National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

University of Alberta

Integration Approach to Process Modelling Using Neural Network and to Control System Design

by

Heon Chang Kim

A thesis

submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of Master of Science

in

PROCESS CONTROL

Department of Chemical Engineering

Edmonton, Alberta

Fall 1993

National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-88209-3

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Heon Chang Kim

TITLE OF THESIS: Integration Approach to Process Modelling Using Neural Network and to Control System Design

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1993

Permission is hereby granted to the UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purpose only.

The author reserves all other publication rights and other rights in association with the copyright of the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) _____

Permanent Address:
1-383 Hannam-Dong Yongsan-Gu
Seoul, Korea

Date: 05/10/93

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have been read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled Integration Approach to Process Modelling using Neural Network and to Control System Design submitted by Heon Chang Kim in partial fulfillment of the requirements for the degree of Master of Science in PROCESS CONTROL.

Dr. M. Rao

Dr. S. L. Shah

Dr. V. G. Gourishankar

Date: Sept. 10, 93

Abstract

As the modern industrial process is becoming increasingly complex, it is difficult to operate the process effectively. Accordingly, the importance of integrating different tools within a package for intelligent process operation has been realized.

This thesis study primarily consists of two part. A unique feature of this research is the utilization of the meta-system concept for the integration issues both in AI systems and in conventional programs.

In the first part of this study, a multilayer feedforward Backpropagation (BP) neural network has been developed for inferential process modelling, then integrated to an Integrated Distributed Intelligent System (IDIS). To demonstrate the ability of Artificial Neural Network (ANN) in process modelling, ANN was applied to the maximization of the desired product yield by predicting its volatility in refinery plants. Two case studies have been carried out for different refinery processes using the BP network. Due to major drawbacks in the Generalized Descent (GD) method, which is a typical optimization algorithm in BP, the Conjugate Gradient (CG) method was also considered in training ANN. In the first case study, the ANN and Regression Analysis (RA) models were compared in representing the relation between the plant stream data and the product volatility.

In the second case study, ANN model was also applied to demonstrate its ability to fit noisy plant data. The ANN model can then be used for inferred volatility control.

The second part of the thesis is regarding the development of an interactive Computer-Aided Control System Design (CACSD) environment for chemical processes by applying the meta-system concept to the integration of several independently developed conventional programs. Its result is an interactive graphical software package PCET (Process Control Engineering Teachware). PCET has a hierarchical structure of several independently developed subprograms which are integrated under the control of a supervising system, meta-system. It covers a wide spectrum of process control engineering applications, including Time Domain Analysis (TDA), Routh Stability Criterion (RSC), Root Locus Technique (RLT), Frequency Domain Analysis (FDA), Discrete-time System Analysis (DSA), Linear State Space Analysis (LSA) and Industrial Application Case (IAC). This package can be used to design and analyze control systems. It can also improve understanding of the basics of process control engineering, and help users gain experience on simulation and computer-aided design.

Acknowledgments

I would like to express my appreciation towards Dr. Ming Rao, my advisor, for his strong encouragement and support.

I would also like to thank Mr. H. Qiu, Professor P. Brook (Faculty of Education) and Dr. X. Shen for their important suggestions and comments.

I would like to express my sincere appreciation to all those who made an important contribution on the development of several independent subprograms, Dr. M. Rao and Mr. R. Dong for Time Domain Analysis (TDA), Mr. Y. Ying for Routh Stability Criterion (RSC), Ms. H. Zhou for Root Locus Technique (RLT) and Mr. M. Sardagar for Linear State Space Analysis (LSA).

I would like to express my gratitude to ESSO Petroleum Canada Ltd., Strathcona Refinery, for providing an industrial environment for my research.

Last, but not least, I would like to thank all members of my committee for taking the time to advise me on my thesis:

Dr. Hayes, Chairman

Dr. Shah, Committee Member

Dr. Gourishankar, Committee Member

Dr. Rao, Committee Member and Supervisor.

My thesis research is financially supported by Canadian Pulp and Paper Association, Daishowa Peace River Pulp Mill, Imperial Oil limited University Research Grants, J & K International, Natural Science and Engineering Research Council of Canada (NSERC), Proctor & Gamble Cellulose at Grande Prairie, The Faculty of Graduate Studies and Research and The University of Alberta Teaching Research Fund.

Table of Contents

Abstract i
Acknowledgments iii
Table of Contents v
List of Tables viiii
List of Figures x

Chapter 1:
Introduction 1
1.1 Motivation 1
1.2 Objectives and Scope 6
1.3 Organization 8

Chapter 2:
Inferential Process Modelling via Artificial Neural Network 10
2.1 Introduction 11
2.2 Background on Artificial Neural Network 13
2.2.1 What is an Artificial Neural Network? 13
2.2.2 History 14
2.2.3 Biological and Artificial Neural Network 16
2.2.4 Architecture 18
2.2.5 Backpropagation Learning Paradigm 22
2.2.6 Generalized Descent (GD) Method 23
2.3 Case Studies for Refinery Process 26
2.3.1 Process Background 27
2.3.2 Case Study I 29
2.3.2.1 Regression Analysis Model 30
2.3.2.2 Artificial Neural Network Model 33

2.3.2.3 Results and Discussion 34
2.3.3 Case Study II 36
2.3.3.1 Conjugate Gradient (CG) Method 36
2.3.3.2 Results and Discussion 37
2.4 Conclusions 38

Chapter 3:
Integrating Artificial Neural Network to IDIS 41
3.1 Integrated Distributed Intelligent System (IDIS) 41
3.2 Meta-system 47
3.2.1 Functions 48
3.2.2 Configuration 50
3.2.2.1 Interface to External Environment 50
3.2.2.2 Meta-Knowledgebase 50
3.2.2.3 Global Database 52
3.2.2.4 Inference Mechanism 53
3.2.2.5 Static Blackboard 53
3.2.2.6 Interface to Internal Subsystems 54
3.3 Meta-COOP 54
3.4 Integration of Neural Network into IDIS 60
3.4.1 Interpreting Connection Weights 62
3.4.2 Implementation 63

Chapter 4:
Integrated Process Control System Design 68
4.1 Introduction 69
4.2 System Construction 71
4.2.1 Integration and Management of CACSD Facilities 71
4.2.2 Integrated Control System Design Environment 73
4.2.2.1 User-friendly Interface 73
4.2.2.2 Graphic Display Capability 75
4.2.2.3 Integrated Distributed System Configuration 76

4.2.2.4 Industrial Process Control Application Case 79
4.3 Functions of PCET 80
4.4 System Evaluation 82
4.5 Conclusions 84

Chapter 5:
Process Control Engineering Teachware 86
5.1 Time Domain Analysis (TDA) 86
5.1.1 Step Input 86
5.1.2 Transfer Functions 88
5.1.3 PID Controller 89
5.1.4 Time Delay 90
5.2 Routh Stability Criterion (RSC) 98
5.2.1 Characteristic Equation and Stability 101
5.2.2 Routh Array 101
5.3 Root Locus Technique (RLT) 108
5.3.1 Starting and End Points 109
5.3.2 Points at Imaginary Axis 109
5.3.3 System Analysis and Design 110
5.4 Frequency Domain Analysis (FDA) 115
5.4.1 Sinusoidal Signal 115
5.4.2 Frequency Response 116
5.4.3 Nyquist Plot 116
5.4.4 Bode Diagram 117
5.5 Discrete-time System Analysis (DSA) 127
5.5.1 Zero-order Hold 127
5.5.2 Discrete-time System Responses 128
5.5.3 Digital PID Controller 129
5.6 Linear State Space Analysis (LSA) 135
5.6.1 State Space 135
5.6.2 State Space Representation 136
5.6.3 Controllability and Observability 137

5.6.3.1 Controllability Criterion 137
5.6.3.2 Observability Criterion 138
5.6.4 Diagonalization of Matrices 138
5.7 Industrial Application Case (IAC) 142
5.7.1 Cascade Control 147

Chapter 6:
General Discussion and Conclusions 156
6.1 Results 156
6.2 Contributions 158

Bibliography 160

Appendices
A Source Code for ANN 167
B Meta-knowledgebase Source Code for Integration of ANN and CWI into IDIS 184
C PCET Evaluation Form 188

List of Tables

Table 4.3.1 PCET functions 83

Table 5.1.1 Functions of TDA in PCET 97

Table 5.2.1 Functions of RSC in PCET 105

Table 5.3.1 Functions of RLT in PCET 112

Table 5.4.1 Functions of FDA in PCET 121

Table 5.5.1 Functions of DSA in PCET 132

Table 5.6.1 Functions of LSA in PCET 141

Table 5.7.1 Functions of IAC in PCET 149

List of Figures

Figure 1.1 Qualitative and quantitative analysis 2

Figure 1.2 Software architecture of coupling intelligent system 5

Figure 2.2.3.1 Biological neuron 17

Figure 2.2.3.2 Artificial neuron 19

Figure 2.2.3.3 Continuous sigmoidal activation function 19

Figure 2.2.4.1 Three layer feedforward network architecture 20

Figure 2.2.6.1 Backpropagation learning paradign 24

Figure 2.3.1.1 Typical refinery unit 28

Figure 2.3.1.2 Distillation curve 31

Figure 2.3.2.3(a) Comparison of ANN model and RA model on fitting the data (Case Study I) 35

Figure 2.3.2.3(b) Prediction of volatility (Case Study I) 35

Figure 2.3.3.2(a) Fitting noisy plant stream data (Case Study I) 39

Figure 2.3.3.2(b) Prediction of volatility (Case Study II) 39

Figure 3.1 Integrated distributed intelligent system structure 46

Figure 3.2 Meta-system configuration 51

Figure 3.3 Integration of inference strategies in Meta-COOP 58

Figure 3.4.2.1 Communication between meta-system and subsystems 64

Figure 3.4.2.2 Indirect communication procedure 64

Figure 3.4.2.3 Data flow diagram for ANN, CWI and any other subsystem under the supervision of meta-system 67

Figure 4.2.1 Overall hierarchical structure of PCET 78

Figure 4.3.1 Main menu of PCET 81

Figure 5.1.1(a) Process model selection of Example 5.1 99

Figure 5.1.1(b) Process parameters input of Example 5.1 99

Figure 5.1.1(c) PID controller parameters input of Example 5.1 100

Figure 5.1.1(d) Time domain response of Example 5.1 100

Figure 5.2.1(a) Characteristic equation parameters of Example 5.2 106

Figure 5.2.1(b) Routh array of Example 5.2 106

Figure 5.2.1(c) Roots of the characteristic equation of Example 5.2 107

Figure 5.3.1(a) Open loop transfer function without controller of Example 5.3 113

Figure 5.3.1(b) Root locus with respect to proportional gain of Example 5.3 113

Figure 5.3.1(c) Equivalent transfer function with PI controller of Example 5.3 114

Figure 5.3.1(d) Root locus with respect to t_i without controller of Example 5.3 114

Figure 5.4.1(a) Process model and its parameters of Example 5.4 123

Figure 5.4.1(b) Nyquist plot without controller of Example 5.4 123

Figure 5.4.1(c) Bode diagram without controller of Example 5.4 124

Figure 5.4.1(d) Gain and phase margin with P controller of Example 5.4 124

Figure 5.4.1(e) Gain and phase margin with PI controller of Example 5.4 125

Figure 5.4.1(f) Gain and phase margin with PID controller of Example 5.4 125

Figure 5.4.1(g) Time domain responses with P, PI and PIDcontroller of Example 5.4 126

Figure 5.5.1 Discrete-time system configuration 133

Figure 5.5.1(a) Discrete-time system parameters of Example 5.5 133

Figure 5.5.1(b) Digital PID controller parameters of Example 5.5 134

Figure 5.5.1(c) Discrete-time response of Example 5.5 134

Figure 5.6.1(a) System dimension of Example 5.6 143

Figure 5.6.1(b) System parameters of Example 5.6 143

Figure 5.6.1(c) Controllability matrix of Example 5.6 144

Figure 5.6.1(d) Observability matrix of Example 5.2 144

Figure 5.6.1(e) Controllability and observability analysis of Example 5.3 145

Figure 5.6.1(f) Modal matrix of Example 5.3 145

Figure 5.6.1(g) Diagonalized system of Example 5.3 146

Figure 5.7.1(a) Headbox control system information in IAC 152

Figure 5.7.1(b) Cascade control system configuration in IAC 152

Figure 5.7.1(c) Nyquist plot without controller of Example 5.4 153

Figure 5.7.1(d) Secondary loop controller tuning information of Example 5.7 153

Figure 5.7.1(e) Secondary loop response of Example 5.7 154

Figure 5.7.1(f) Primary loop controller tuning information of Example 5.7 154

Figure 5.7.1(g) Primary loop response of Example 5.7 155

Chapter 1

Introduction

This chapter describes the motivation and scope of this research as well as the organization of the thesis.

1.1 Motivation

Many existing symbolic reasoning systems are developed for specific purposes, and production rules are used to represent domain expertise. These systems can only process symbolic information and make heuristic inferences. The lack of numerical computation and coordination between single applications limits their capability to solve real engineering problems. Solving engineering problems as shown in Figure 1.1, however, normally requires qualitative and quantitative analy-

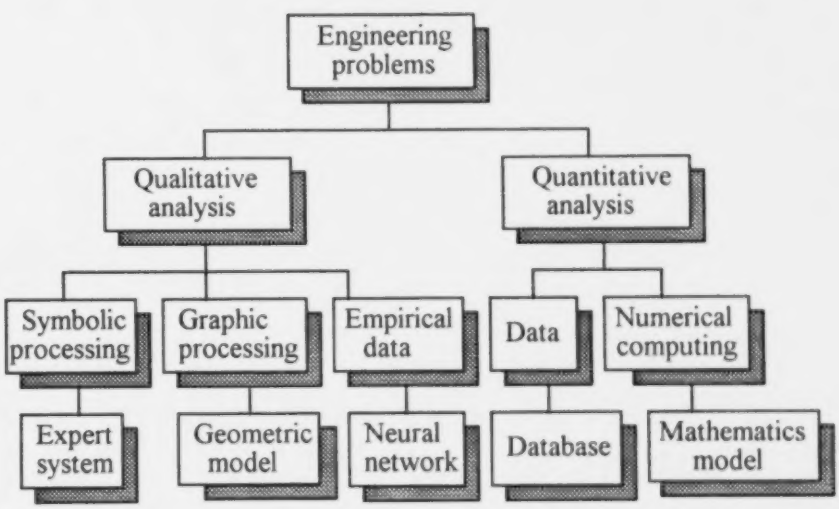


Figure 1.1 Qualitative and quantitative analysis

ses together. Usually, qualitative decisions are mainly based on symbolic and graphical information, while quantitative analysis is more conveniently performed using numerical information. Both methods often complement each other.

Any numerical solution, no matter how perfect it is, is always an approximation to the true solution. The true solution is always represented analytically. Analytical solutions can only be obtained by symbolic processing. Meanwhile, a main disadvantage of the existing symbolic reasoning systems is their inability to handle numerical computation. This makes these systems less useful for many complicated engineering problems. Moreover, as a part of the accumulated knowledge of human expertise, many practical and successful numerical computation packages have been made available. Even if Artificial Intelligence (AI) emphasizes symbolic processing and non-algorithmic inference (Buchanan, 1985), it should be noted that the utilization of numerical computation will make intelligent systems more powerful in dealing with engineering problems. Like many modern developments, AI and its applications should be viewed as a welcome addition to the technology, but they cannot be used as a substitute for numerical computation.

The coordination of symbolic reasoning and numerical computation is essential in developing intelligent systems. More and more, the importance of coordinating symbolic reasoning and numerical computing in knowledge-based systems is being recognized. It has been realized that if applied separately, neither symbolic reasoning nor numeric computing can successfully address all engineering prob-

lems. Complicated problems cannot be solved by purely symbolic or numerical techniques (Jacobstein et al., 1988; Wong et al., 1988).

Close coordination between symbolic reasoning and numerical computation is required in the intelligent manufacturing environment. Figure 1.2 distinguishes the coupling intelligent system from the symbolic reasoning system from the viewpoint of software architecture. So far, many coupling intelligent systems have been developed in various engineering fields to enhance the problem-solving capacity of the existing symbolic reasoning systems (Kitzmiller and Kowalik, 1987). In the intelligent decisionmaker for problem-solving strategy of optimal control (IDSOC), a set of numerical algorithms to compute certainty factors is coupled in the process of symbolic reasoning (Rao et al., 1988). Another coupling intelligent system SFPACK incorporates expert system techniques in design package then supports more functions to designers (Pang et al., 1990). Written in Franz Lisp, CACE-III can control the startup of several numerical routines programmed in FORTRAN (James et al., 1985). Similar consideration was taken into account by Astrom's group (Astrom et al., 1986). More and more, the coordination of symbolic reasoning and numerical computing in knowledge-based systems attracts much attention.

The methods of integrating single symbolic reasoning systems and numerical computation packages were proposed by Kitzmiller and Kowalik (1987). A few developers tried to develop coupling intelligent systems with conventional lan-

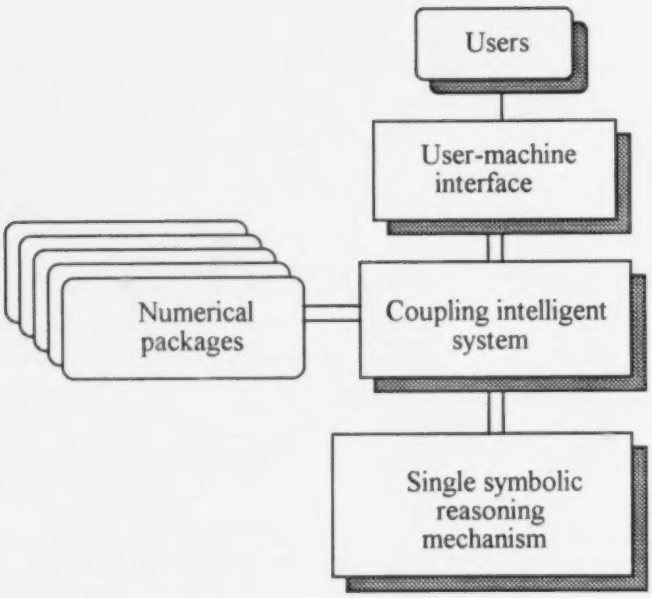


Figure 1.2 Software architecture of coupling intelligent system

guages, such as FORTRAN, so that these symbolic reasoning systems could be used as subroutines in a FORTRAN main program. Others suggested developing coupling intelligent systems in conventional languages in order to achieve the integration of numerical algorithms and symbolic inference. However, these methods prohibit developing and using the individual programs separately. This makes acquiring new programs very difficult, and is not cost-effective.

Numerical languages often have a procedural flavor, in which the program control is command-driven. They are very inefficient when dealing with processing strings. Symbolic languages are more declarative and data-driven. However, symbolic languages are very slow to execute numerical computations. Coupling of symbolic processing with numerical computing is desirable to use numerical and symbolic languages in different portions of a software system.

Currently, not all of the expert system tools or environments provide programming techniques for developing coupling intelligent systems. However, many software engineers are now building the general-purpose tools for coupling intelligent systems that will be beneficial to AI applications to engineering domains.

1.2 Objectives and Scope

In order to solve the problems mentioned above, concepts of Integrated

Distributed Intelligent System (IDIS) were proposed by Rao et al. (1987). In the past several years, the Intelligence Engineering Laboratory at the University of Alberta has been engaged in studying and developing a new architecture to control and manage large-scale intelligent systems for industrial applications. So far, a prototype IDIS platform has been implemented.

IDIS is a large knowledge integration environment, which consists of several symbolic reasoning systems, numerical computation programs, a database management subsystem, computer graphics packages, multimedia interfaces, as well as a meta-system. It makes use of the advanced object-oriented programming technique in C++ language. The integrated software environment allows the running of programs written in different languages, communication among programs, as well as the exchange of data between programs and databases. These isolated expert systems, numerical packages and programs are under the control of a supervising expert system, namely the meta-system. The meta-system manages the selection, coordination, operation and communication of these programs.

The thesis research is divided primarily into two parts. The first part of this study is regarding the integration of AI tools by the meta-system described above. The specific objectives of the first part of the study are

Integrated Process Modelling using Neural Network

- to develop a neural network system for inferential process modelling;

- to integrate the neural network into IDIS under the control of the supervisory intelligent system, namely, meta-system;

The second part of this study is regarding the integration of conventional programs. Even though the meta-system concept was originally developed for the IDIS, it can be still applied to the integration of conventional programs for the development of an integrated Computer-Aided Control System Design (CACSD) package. The specific objectives of the sencond part of the study are

Integrated Control System Design

- to develop several subprograms dealing with different aspects of process control system design;
- to develop an interactive CACSD environment by applying the meta-system concept to the integration of coventional programs.

A unique feature of these two different integrated environments is the utilization of the meta-system concept for the integration issues both in AI systems and in conventional programs.

1.3 Organization

This thesis consists of six chapters. The first is this introductory chapter, which covers the motivation and general objectives of the thesis. The thesis

research is divided primarily into two parts.

The first part of this thesis is covered in Chapters 2 and 3. Chapter 2 describes the application of a multilayer feedforward ANN with backpropagation learning paradigm for inferential process modelling. To verify its performance, two case studies have been carried out to predict the desired product volatility in refineries. Chapter 4 is regarding the integration of the neural network developed in this study, as a subsystem, into IDIS by utilizing a new meta-system structure, Meta-COOP.

The second part of this thesis, described in Chapters 4 and 5, involves developing several subprograms dealing with different aspects of process control system design and integrating them in an interactive environment under the control of supervisory system. Chapter 4 describes the general integration strategy to develop an integrated interactive computer-aided control system design environment for chemical processes. Chapter 5 contains details functions of Process Control Engineering Teachware (PCET), developed in this study, as well as illustrative examples.

Chapter 6 contains some general concluding remarks, and provides suggestions for future research.

Chapter 2

Inferential Process Modelling via Artificial Neural Network*

In this chapter, an Artificial Neural Network (ANN) is applied to predict the volatility of the product for the maximization of the product yield in refinery plants. Two case studies are carried out for different refinery processes using the multilayer feedforward Backpropagation (BP) network. Due to major drawbacks in the Generalized Descent (GD) method, which is a typical optimization algorithm in BP, the Conjugated Gradient (CG) method is also considered in training ANN. In the first case study, the ANN and Regression Analysis (RA) models are compared in representing the relation between the plant stream data and the product volatility.

---

* Two versions of this chapter have been presented at the 2nd IFAC Workshop on Algorithms and Architectures for Real-Time Control, August 31 - September 2, 1992 in Seoul, Korea and at the 42nd Annual Canadian Chemical Engineering Conference, October 18-21, 1992 in Toronto, Ontario, Canada.

In the second case study, the ANN model is also applied to demonstrate its ability to fit noisy plant data.

2.1 Introduction

Since the early 1980's, there has been an explosive growth in pure and applied research related to ANNs. In recent years, ANN technology has attracted attention in the control community because there are many systems, for which rigorous mathematical models do not exist or are not sufficient to meet a full-scale identification requirement, such as highly nonlinear chemical processes or very complex industrial plants. In fact, there exist very complex systems with unknown characteristics and uncertainties that are difficult to identify using mathematical models. To model such systems, ANNs may be able to provide some promising solutions.

ANNs have proven effective at solving pattern recognition problems in a wide variety of areas including image processing, speech recognition, sensor interpretation, motor control, and system identification (Haesloop and Holt, 1990). Several factors motivate the use of ANN models:

1. general mapping capabilities
2. reliance on the knowledge specified through learning instead of a pre-specified algorithm

ability to use vast amounts sensory information

3. capability to respond at high speed to sensory inputs due to its parallel
4. structure

   greater degree of robustness and fault tolerance due to their distributed
5. representation.

Many areas of chemical engineering such as fault detection and diagnosis (Watanabe et al., 1989), process control (Cooper et al., 1992, Donat et al., 1990, and Hernandez and Arkun, 1990), process design, and process modeling and simulation can take advantage of these factors to organize and detect features from unpredictable and/or imprecise process data. For many chemical engineering systems that are difficult to model, a large amount of process measurement data is usually stored in the database. Thus, these stored historical data can be used to develop an ANN model that describes the input and output behavior of the process. ANN can learn and adapt themselves to inputs from actual processes, thus allowing representation of complex engineering systems that are difficult to model with traditional physical engineering relations. In addition to the self-organizing capability of ANN, another advantage is the parallelism inherent in the neurocomputing architecture. This allows ANN system to be implemented using highly parallel hardware to achieve real-time performance.

It is common to model a process prior to implement a modern control algorithm (Haesloop and Holt, 1990). Process modelling can be done by applying

first principles, such as mass and energy balance, which represent process variable interactions from physical consideration. For complex chemical process systems or industrial processes, however, it is often done by applying an identification technique. The identified model is usually a linearized one around a chosen operating condition even for a nonlinear process. Therefore, if the operating condition is altered, the discrepancy between linearized model and true nonlinear system behavior would be significant. ANNs offer the opportunity to directly model nonlinear processes. With a more accurate nonlinear model, the plant-model divergence mentioned above may be reduced. ANN modelling is such an area with a non-programmed adaptive approach to processing engineering data systems. ANNs can adaptively develop transformations in response to their environment. In contrast to programmed computing, ANNs are able to develop a mapping function from examples of that function's operation.

2.2 Background on Artificial Neural Network

This section provides fundamentals of ANNs including the multilayer feedforward backpropagation learning paradigm.

2.2.1 What is an Artificial Neural Network?

ANN is nothing more than a computational system that performs brainlike

functions simply because it was modelled similar to the human brain. It is composed of highly interconnected simple processing elements, called artificial neurons, in parallel. They are organized in patterns similar to biological neural network. Due to its structural and functional resemblance to biological neural network, it exhibits a number of characteristics of the human brain, for examples, learning from experience by modifying its behavior in response to its environment, generalizing from previous examples to new ones as a result of its structure and abstracting essential characteristics of a set of inputs containing irrelevant data.

2.2.2 History

The idea of an ANN was originally conceived as an attempt to model the biophysiology of the human brain, in other words, to understand and explain how the human brain operates and functions. Along with the progress in neuroanatomy and neurophysiology, psychologists were developing models of human learning in order to produce computational systems that perform brainlike function.

In 1949, Hebb proposed a learning law that proved to be the most successful model and became the starting point for ANN training algorithms. It showed scientists how a network of neurons could exhibit learning behavior.

In the 1950s and 1960s, a group of researchers combined these biological

and psychological insights to produce the first ANN, which was initially implemented as an electronic circuit. Later it was converted to the more flexible medium of computer simulation. Then many researchers including Marvin Minsky developed networks consisting of a single layer of artificial neurons call perceptrons and applied them to such diverse problems as weather prediction, electrocardiogram analysis, artificial vision, etc. At that time, they thought that reproducing the human brain was only a matter of constructing a large enough network. However, networks failed to solve problems superficially similar to those they had been successful in solving. So Minsky, who is a respected senior scientist in this field, carefully applied mathematical techniques and developed rigorous theorems regarding network operation.

In 1969, Minsky published a book titled PERCEPTRONS. In this book, he proved that the single-layer networks can not theoretically solve many simple problems, including the function performed by a simple exclusive-or gate. Even he was not optimistic about the potential for progress. So discouraged researchers left the field for areas of great promise. Government agencies redirected their funding and ANN lapsed into obscurity for nearly two decades. Nevertheless, a few dedicated scientists continued their efforts. Gradually, a theoretical foundation emerged, upon which the more powerful multilayer networks of today are being constructed.

In 1987, Rumelhart and other researchers invented backpropagation which

provides a systematic means for training multilayer network, thereby overcoming limitations presented by Minsky. Although a broad range of ANN architectures and learning paradigms are available, the backpropagation algorithm for multilayer feedforward network is the most popular approach for current engineering problems because it is a simple and powerful method.

2.2.3 Biological and Artificial Neural Network

A human brain is known to contain over a hundred billion neurons which are considered to be computing elements. These neurons communicate throughout the body by way of nerve fibers that make perhaps one hundred trillion connections. Figure 2.2.3.1 shows the structure of a pair of typical biological neurons. A neuron is the fundamental building block of the nervous system. It is called a cell similar to all cells in the body; however, certain critical specializations allow it to perform all of the computational and communicational functions within the brain. The neuron consists of three sections: the cell body, the dendrites, and the axon, each with separate but complementary functions. Dendrites extend from the cell body to other neurons where they receive signals at a connection point called a synapse. On the receiving side of the synapse, these inputs are conducted to the cell body. There they are summed, some inputs tending to excite the cell, others tending to inhibit its firing. When the cumulative excitation in the cell body exceeds a thresh hold, the cell fires, sending a signal down the axon to other neurons. Even though this basic functional outline has many complexities and

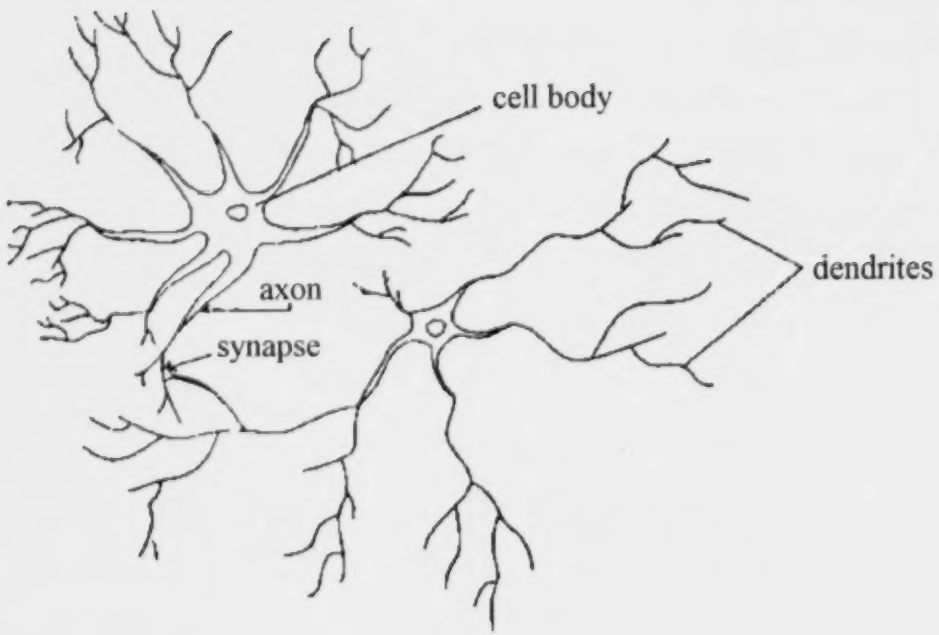


Figure 2.2.3.1 Biological neuron

exceptions, most ANNs model only these simple characteristics.

The artificial neuron was designed to mimic the first-order characteristics of the biological neuron. Figure 2.2.3.2 represents an artificial neuron model. In essence, a set of inputs labeled X_1, X_2, ..., X_n is applied to the artificial neuron. Each input represents the output of another neuron. Each signal is multiplied by a corresponding weight W_1, W_2, ..., W_n, before it is applied to the summation block, labeled Σ. Each weight corresponds to the strength of a single biological synaptic connection. The summation block, corresponding roughly to the biological cell body, adds all of the weighted inputs algebraically to determine the activation level of the neuron. The summed signal S is usually further processed by an activation function F to produce the neuron's output signal Y. If the activation function F compresses the range of S, so that Y never exceeds some low limits regardless of the value of S, F is called a squashing function. The squashing function is often chosen to be sigmoidal function that is continuous and strictly monotonic as shown in Figure 2.2.3.3. This function can be expressed mathematically as $F(S)=1/(1+e^{-S})$. Another commonly used activation function is the hyperbolic tangent, $F(S)=\tanh(S)$.

2.2.4 Architecture

A typical representation of the three-layer feedforward network is shown in Figure 2.2.4.1. It is composed of many interconnected processing units or neurons

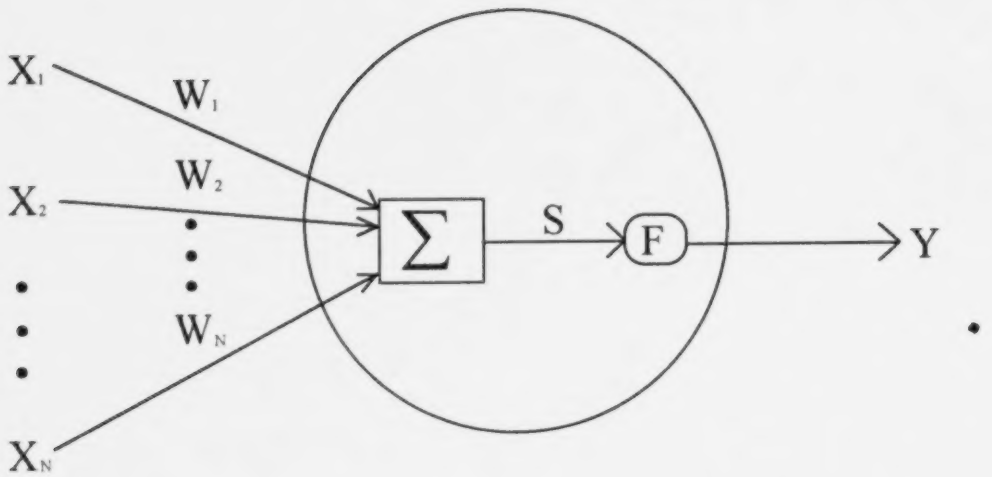


Figure 2.2.3.2 Artificial neuron

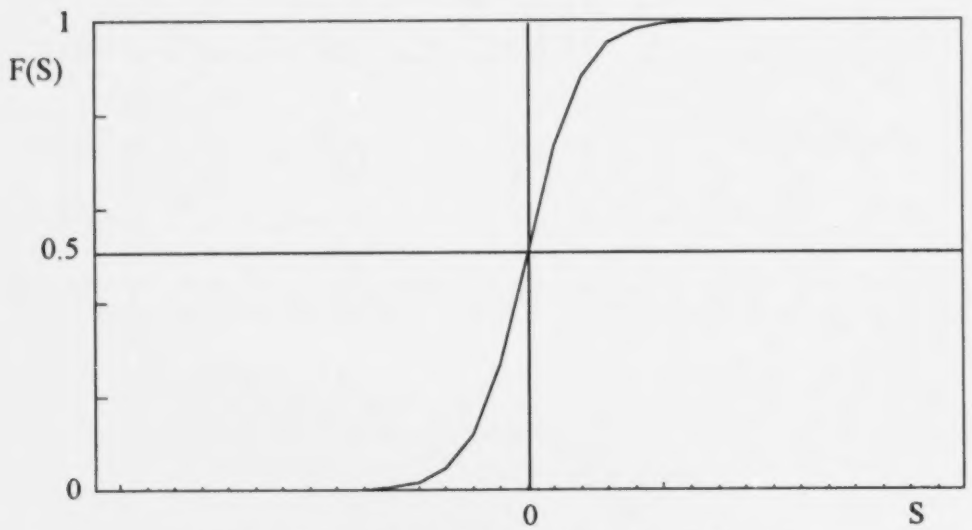
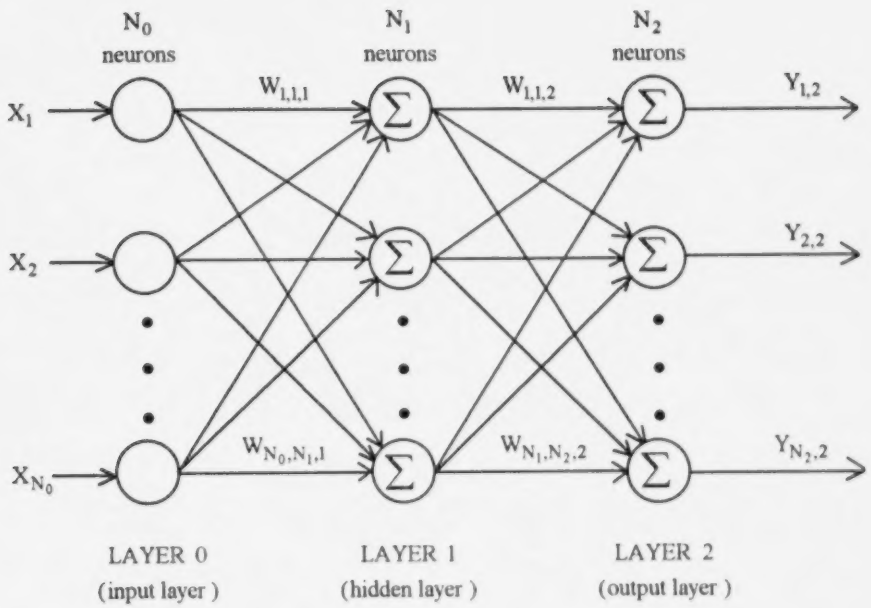


Figure 2.2.3.3 Continuous sigmoidal activation function



$$S_{j,k} = \sum_{i=1}^{N_{k-1}} W_{i,j,k} X_i$$

$$Y_{j,k} = F(S_{j,k})$$

F(x)=activation function

Figure 2.2.4.1 Three layer feedforward network architecture

(N_0 neurons in the input layer, N_1 neurons in the hidden layer and N_2 neurons in the output layer) organized in successive layers. The analysis and design of any multilayer neural networks can be done similarly. The neuron j in the layer k first computes the weighted sum of the N_{k-1} inputs,

$$S_{i,j,k} = \sum_{i=1}^{N_{k-1}} W_{i,j,k} X_i \qquad (2.2.4.1)$$

where X's and W's represent the given input and its associated weight, respectively. Then it outputs a nonlinear function of the sum in (2.2.4.1), which serves as an input to the next layer.

$$Y_{j,k} = F(S_{j,k}) \qquad (2.2.4.2)$$

Every neurons except those in the input layer functions in the same way. In the first or input layer, neurons do not perform any computation but simply distribute their inputs to all neurons in the next layer.

Only unknowns in this representation are these associated weights between layers. So, in order to use the neural network model, these weights should be estimated first. The evaluation of appropriate weights is called learning or training. The objective of learning is to train the network so that application of a set of inputs produces the desired or at least consistent set of outputs.

2.2.5 Backpropagation Learning Paradigm

Even though a broad range of ANN architectures and learning paradigms are crrently available, the BP algorithm for multilayer feedforward ANNs is the most common approach for current engineering applications mainly because it is a simple and powerful method.

In the BP learning paradigm, estimation of weights is accomplished by sequentially applying a set of inputs and desired outputs to the network, while adjusting network weights according to a predetermined procedure. During training, the network weights gradually converge to values such that each input data set produces the desired output.

The objective of the BP learning paradigm is to minimize the overall error, E, between the desired and actual output:

$$E = \frac{1}{2}\sum_{l=1}^{p}\sum_{m=1}^{N_2}(Y_{m,2}^{(l)} - D_m^{(l)})^2 \qquad (2.2.5.1)$$

where D's and Y's represent the desired outputs and the actual outputs, respectively. p represent the number of data set used to train the network. The sum of errors between the desired outputs and actual outputs is required to be smaller than the given error tolerance for the network to identify the system using the input-output data.

BP is known as a supervised synchronous learning paradigm. It is supervised because for every input presented for learning, the expected or correct corresponding output is available so the network can modify its weights accordingly by applying the gradient descent method. It is synchronous because all the weights are modified at each learning step. The network is expected to learn a number of data patterns composed of inputs and associated outputs.

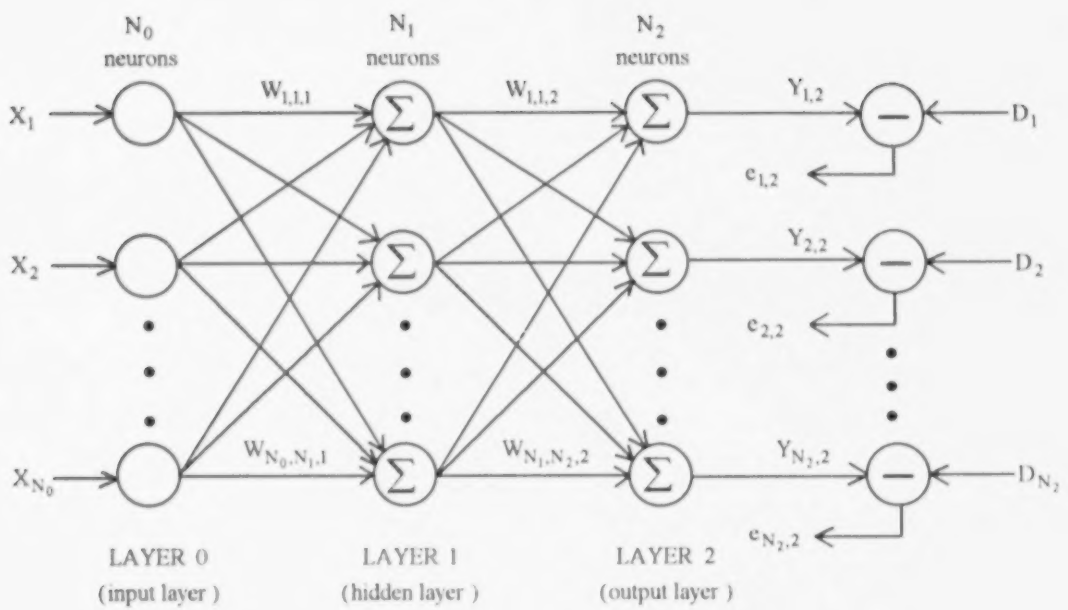
2.2.6 Generalized Descent (GD) Method

The basic strategy behind the standard BP algorithm is to perform gradient descent in parameter space based on the individual errors between a set of network mappings and the corresponding set of desired mapping examples.

$$E_{GD} = \frac{1}{2}\sum_{i=1}^{N_2}(Y_{i,2} - D_i)^2 \qquad (2.2.6.1)$$

Before starting the training process, all of the weights are initialized at small random values. This ensures that the network is not saturated by large values of the weights, and prevents certain other training pathologies. The first pattern is then presented to the network. Figure 2.2.6.1 shows the training process. Learning takes place in two successive steps: forward and backward passes.

During the forward pass, as described in the previous section, each neuron



$$E = \frac{1}{2} \sum_{l=1}^{p} \sum_{m=1}^{N_2} (D_m^{(l)} - Y_{m,2}^{(l)})^2$$

Figure 2.2.6.1 Backpropagation learning paradigm

i in layer j computes the weighted sum of its inputs $S_{i,j}$ and outputs $Y_{i,j}$ as in (2.2.4.1) and (2.2.4.2), respectively. This process is called forward because it takes place in layer j-1 before layer j and therefore propagates from the input layer to the output layer. At this point the activations or outputs of all the neurons of the network are available.

During the backward pass, the output error is backpropagated in the reverse direction to minimize the error, $E_{i,2}$, between the desired and actual output,

$$E_{i,2} = (Y_{i,2} - D_i) F'(S_{i,2}) \qquad \text{for } 1 \le i \le N_2 \text{ (output layer)} \qquad (2.2.6.2)$$

where F' denotes the derivative of the activation function. For each of the N_1 hidden units the error is computed as,

$$E_{i,1} = F'(S_{i,1}) \sum_{k=1}^{N_2} E_{k,2} W_{i-k,1} \qquad \text{for } 1 \le i \le N_1 \text{ (hidden layer)} \qquad (2.2.6.3)$$

Once the errors has been computed, the weights are modified as,

$$W_{i-k,j} = W_{i-k,j} + \eta E_{k,j} Y_{i,j} \qquad (2.2.6.4)$$

where η is the step size of the gradient method and is sometimes called the learning rate. Therefore, the weight matrix between the output and hidden layers

can be updated from the output layer to the hidden layer. The next pattern is then presented and the calculations are repeated.

The training procedures are iteratively carried out until the ANN can represent the relationships of these data within the given error tolerance. The non-parametric model is obtained by the fully connected structure and values of the weights. If well trained, the neural network can identify the system in terms of the input-output data.

2.3 Case Studies for Refinery Process

From an engineering point of view, ANN modelling involves unknown parameters (the weight values) and given network topology (units, layers and connections). It can be referred to as a non-parametric model that is totally different from conventional mathematical models. By this definitions, ANNs can be used to represent the input-output data and relationships of any physical system, as long as sufficient input-output system data are collected. It is also assumed that ANN modelling would only be used for systems whose mathematical models cannot be obtained. Otherwise, conventional identification methods are probably superior.

The new concept of non-parametric identification is based on the properties of ANNs using the given topology and adjustable weights. The main advantages

from introducing ANNs are derived from the utilization of their learning ability, associative memory, pattern classification and other such features.

2.3.1 Process Background

Refinery units usually process a combination of several different kinds of crudes, to produce several products at different stages. Figure 2.3.1.1 is a schematic diagram of a typical refinery unit. The objective of process control on these processes is to maximize the desired product yield while its volatility is kept within specification. Stage temperature, which has its primary impact on the desired product volatility, is adjusted on a basis of the desired product viscosity; on the other hand, laboratory measurement of the desired product volatility is used as a basis to adjust the drawoff rate or yield of the desired product, which primarily affects its viscosity. However, since the processes are very complex and exhibit high nonlinearity, in which there exist a large number of variables affecting the desired product yields, the desired product yield is maximized by monitoring its volatility. In other word, maximum desired product yield is achieved by maximizing its volatility within specification.

In practice, to maximize the desired product yield, its volatility is controlled by the following control strategy:

1. fixing the desired product yield by operator entered target

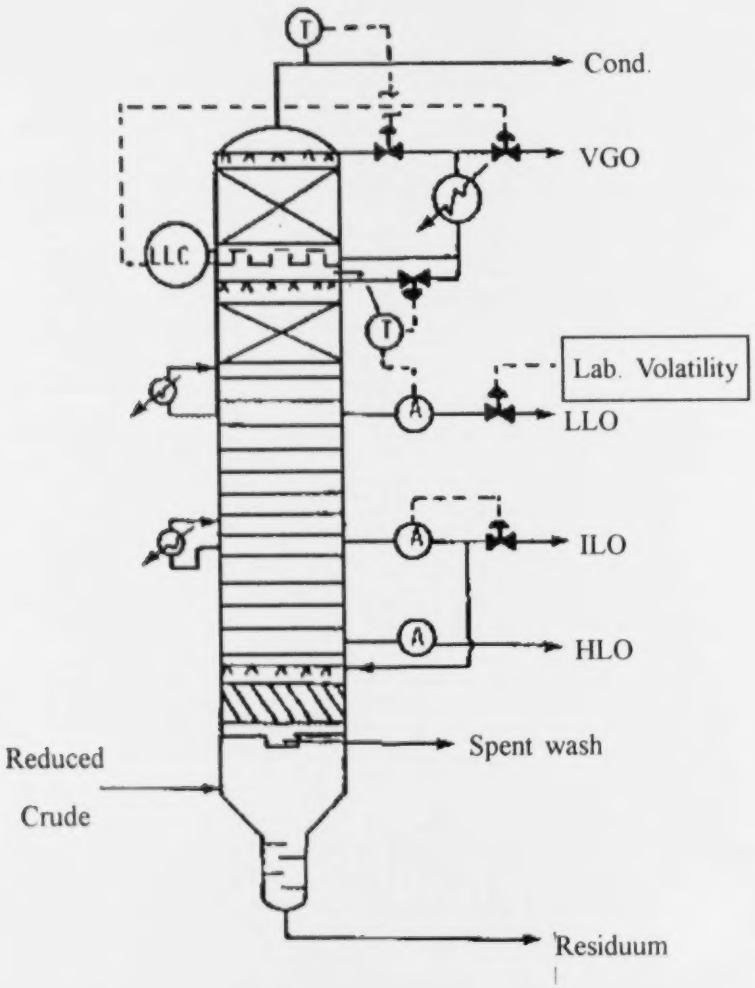


Figure 2.3.1.1 Typical refinery unit

2. controlling the desired product viscosity by manipulating the other product draw rate
3. resetting the desired product yield target on the basis of laboratory measurement of the desired product volatility

There are several problems in this control strategy. The first is that the yield target is reset on the basis of the desired product volatility normally once a day. It almost always results in volatility giveaway (below or above specification). Volatility below specification indicates a lost opportunity for higher yield. Volatility above specification is off-specification product: hence a deterrent to raising the yield target, then the controller tends to operate with the conservative yield target of the desired product, which may also result in volatility giveaway. Another problem is that since the desired product volatility is measured infrequently via off-line methods in the quality control laboratory, it takes a long time to understand the full impact of manipulated variables on the volatility. Therefore, maximization of the desired product yield can not be entirely satisfied by this control strategy.

2.3.2 Case Study I

To reduce the volatility giveaway, an on-line volatility analyzer can be used. The use of the volatility analyzer, however, requires a high maintenance, for examples, frequent recalibration, regular decoking, etc. The alternative is controlling the volatility as inferred from tower data. The operator can then enter a

volatility target directly. While satisfactory performance can be expected from a very well maintained volatility analyzer, the inferred volatility scheme is easily justified based on the low development cost. Therefore, use of a volatility analyzer could be avoided if the volatility can be predicted reliably from stream quality data.

From the distillation curve shown in Figure 2.3.1.2, where viscosity can be seen to depict the midpoint of the cut, a few observations can be made. First, some inverse relationship exists between viscosity of the cut and its volatility. For a given cut width, an increase in viscosity implies a decrease in volatility (case B). Second, for a given midpoint of the cut, a varying cut width affects volatility: a wider cut will have higher volatility and vice versa (case C). Third, the shape of cut's light end tail affects the cut's volatility. From these observations, the following empirical relationship can be postulated:

Volatility = Function (viscosity, stage temperature, distillation)

where the stage temperature is chosen to indicate the cut width and distillation is factored in as an internal reflux ratio.

2.3.2.1 Regression Model

In the Sarnia Refinery, laboratory measurements of the desired product have been correlated to stream quality data by the linear regression technique. Because

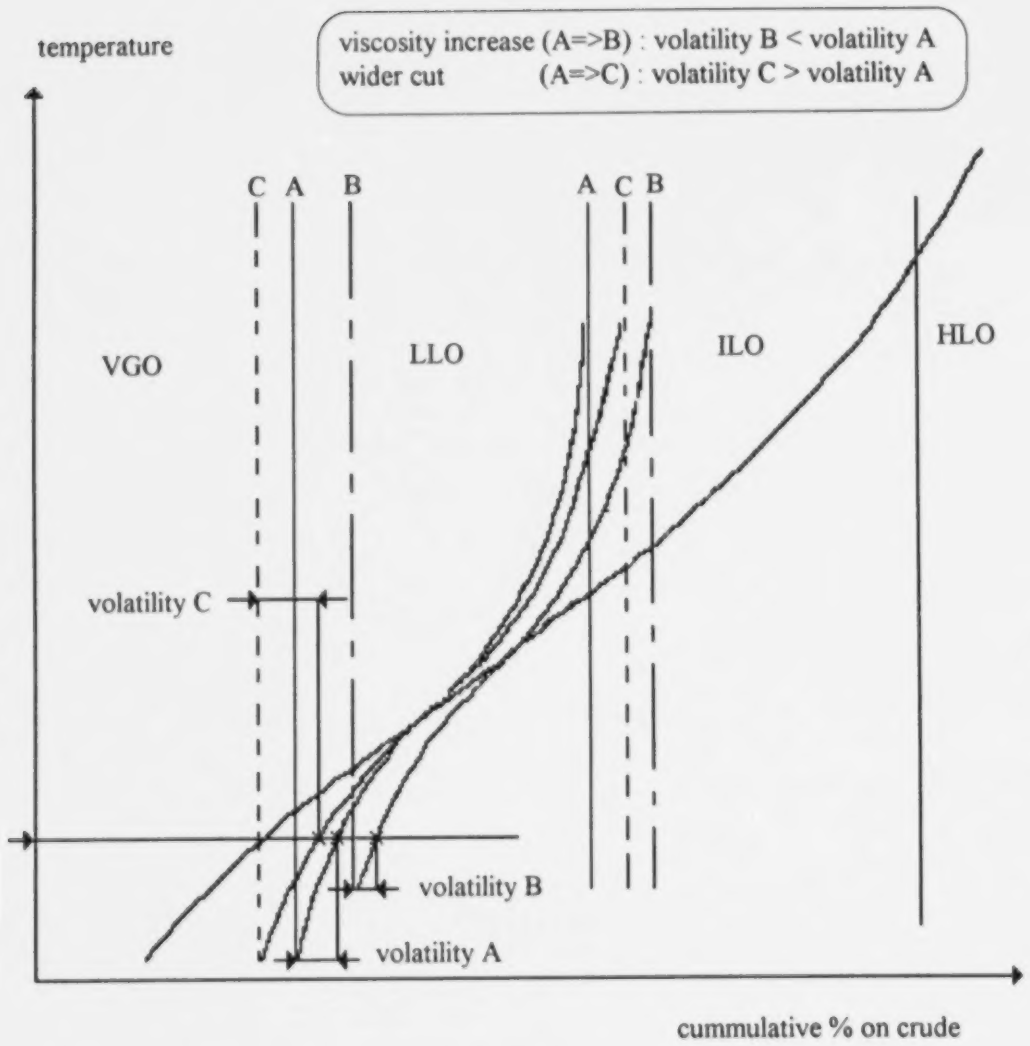


Figure 2.3.1.2 Distillation curve

of difficulty in correlating a time-stamped laboratory result to noisy plant data (error in the dependent variable plays havoc with traditional statistical methods), they used the data from an ASPECT simulation of the process to determine the functional relationship. The real data was then used to adjust the parameters of the regression equation. The viscosity measurement in real data was the same as in the simulation.

Because of significant cross-correlation between two independent variables (stage temperature and viscosity), normalization of stage temperature vs viscosity was made. The final correlation was established between the desired product volatility and its viscosity, stage temperature and the internal reflux ratio above the desired product draw tray.

$$Y=a_0+a_1X_1+a_2X_1^2+a_3X_1^3+a_4X_2+a_5X_3+a_6X_3^2 \qquad (2.3.2.1.1)$$

where

Y = the volatility of the desired product

X_1 = the 'quality A' of the desired product

$X_2 = (\alpha_1-\beta_1)/b_1$

$X_3 = (\alpha_2-\beta_2)/b_2$

α_1 = the 'variable 1'

β_2 = the 'variable 2'

$b_1 = a_7 + a_8 X_1$

$b_2 = a_9 + a_{10} X_1$

$a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}$ = constants

2.3.2.2 Artificial Neural Network Model

As pointed out in Section 2.2.6, standard backpropagation based on steepest descent algorithm is very slow to converge because of the constant learning rate. In order to increase the learning speed of the ANN, a new learning rate function $\eta(t)$ can be introduced as

$$\eta(t) = \begin{cases} \lambda_0 & \left\|\Delta W_{i,j,k}\right\|^2 \geq \delta \\ \lambda + \lambda_0 e^{-\mu t} & \left\|\Delta W_{i,j,k}\right\|^2 < \delta \end{cases} \qquad (2.3.2.2.1)$$

where λ and λ_0 are parameters to be chosen, and are very small ($\lambda_0 > \lambda > 0$, μ and $\delta > 0$). They are system dependent and usually determined by experience and knowledge on neural networks.

In the initial stage, $\eta(t)$ has a relatively larger value because λ_0 is chosen bigger, $\lambda_0 > \lambda$, in order to promote a fast learning convergence. When the connection weight matrix is convergent to a certain degree ($\|\Delta W_{i,j,k}\|^2 < \delta$), $\eta(t)$ starts decreasing until $\eta(t) = \lambda$, and λ is selected relatively small for better convergence in the final stage.

The ANN selected for the prediction of the desired product volatility from the stream data has 3 neurons in the input layer (N_0=3) with 5 neurons in the single hidden layer (N_1=5) and 1 neuron in the output layer (N_2=1). The data set used for learning is the same as that used for regression analysis at the Sarnia refinery. λ_0, λ, and μ were 0.5, 0.09 and 1, respectively.

2.3.2.3 Results and Discussion

A comparison of the performance between the RA model and the ANN model for fitting the data is shown in Figure 2.3.2.3(a). The results show that the ANN model clearly outperforms the regression model for all points. With the data set not used for ANN learning, the ability of the ANN model to predict volatility is also investigated. Figure 2.3.2.3(b) shows a comparison of RA model and ANN model predictions for the desired product volatility from the tower stream data. The results show that the ANN model also accommodates all 9 points much better than the RA model.

Long term changes in operating conditions or change crude composition will affect the accuracy of the ANN model as well as the regression model. Some maintenance of the correlation is expected (i.e. periodically retraining ANN for more accurate prediction of volatility or including crude composition changes in ANN learning input variables).

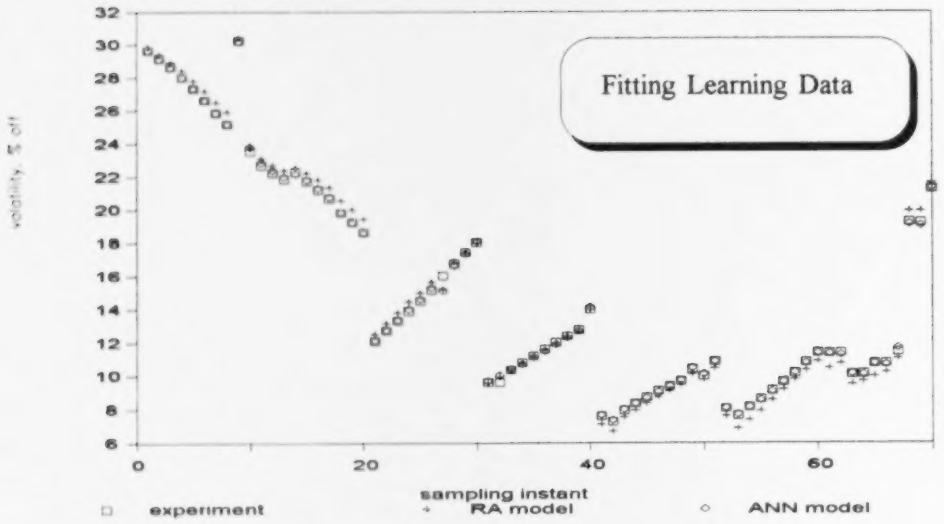


Figure 2.3.2.3(a) Comparison of ANN model and RA model on fitting the data (Case Study I)

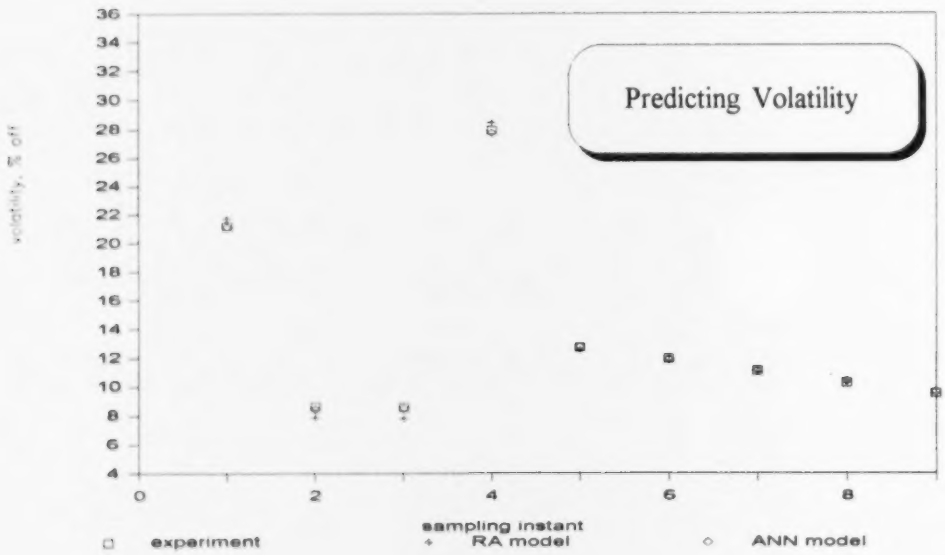


Figure 2.3.2.3(b) Prediction of volatility (Case Study I)

2.3.3 Case Study II

With the experience described in Case Study I, neural network approach for inferential modelling has been applied to ESSO Strathcona Refinery. They also process several other kinds of crudes. In this case study, more variables were considered to cope with reality. For example, feed composition may vary from time to time. Again, due to difficulty in dealing with noisy plant data by regression analysis, only neural network model has been evaluated

2.3.3.1 Conjugate Gradient Method

The standard GD method, in minimizing the objective function, is known to have several major drawbacks. First, it adapts the weights based on the error from a single pattern representation instead of the overall error E which is the true objective function to be minimized. Secondly, the rates of convergence of this algorithm are extremely slow and the improvement per iteration falls sharply. Third drawback is that this method often leads to a local minimum.

To overcome the deficiencies of GD method mentioned above, Conjugate Gradient (CG) method can be used to train the network. This method adapts the weights based on the overall error which is the true objective function to be minimized. Therefore it is more robust and has better convergence properties than GD method.

To minimize the overall error E, the CG method updates the weights at n+1th iteration as:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \alpha^{(n)}\,\mathbf{s}^{(n)} \qquad (2.3.3.1.1)$$

where **w** is the vector of weights and α is found by carrying out a unidimensional line minimization in the given search direction **s**. The new search direction at each iteration is found by

$$\mathbf{s}^{(n)} = -\mathbf{g}^{(n)} + \left(\frac{\left\| \mathbf{g}^{(n)} \right\|^2}{\left\| \mathbf{g}^{(n-1)} \right\|^2} \right) \mathbf{s}^{(n-1)} \qquad (2.3.3.1.2)$$

with $\mathbf{s}^{(0)} = -\mathbf{g}^{(0)}$. The elements of the steepest descent vector **g** are given by

Hidden to output layer:

$$g_{i,j} = \sum_{l=1}^{p} (Y_{j,2}^{(l)} - D_j^{(l)}) Y_{j,2}^{(l)} (1 - Y_{j,2}^{(l)}) Y_{i,1}^{(l)} \qquad (2.3.3.1.3)$$

Input to hidden layer:

$$g_{i,j} = \sum_{l=1}^{p} \sum_{m=1}^{N_2} (Y_{m,2}^{(l)} - D_m^{(l)}) Y_{m,2}^{(l)} (1 - Y_{m,2}^{(l)}) Y_{j,1}^{(l)} (1 - Y_{j,1}^{(l)}) W_{i,j,1}^{(l)} X_i^{(l)} \qquad (2.3.3.1.4)$$

2.3.3.2 Results and Discussion

The ANN selected for the prediction of the desired product volatility from

the stream data has 11 units in the input layer (N_0=11) with 13 neurons in the single hidden layer (N_1=13) and 1 neuron in the output layer (N_2=1). The learning data set was collected from the real operating data used for last three years at ESSO Strathcona Refinery. The ability of the ANN model for fitting the real plant data is shown in Figure 2.3.3.2(a). The results show that, to a degree, the ANN model is able to fit even noisy plant data. Figure 2.3.3.2(b) shows the prediction of the desired product volatility from the plant stream data by the ANN model. It also shows that the ANN model can predict the volatility to a certain degree of accuracy.

2.4 Conclusions

In this chapter, it has been demonstrated that ANN can model the product volatility at refinery plants. The ANN model matches the data much better than the RA model. In addition, the ANN model has the ability to fit noisy plant data and to predict the product volatility.

Although ANNs have the capability to cope with the nonlinearity, complexity and uncertainty of dynamic systems in terms of their numerical characteristics in identifying the relationships and features of the system input-output data, there are some of the crucial ANNs problems to be overcome.

ANN needs too much learning time (usually, the smaller acceptable error,

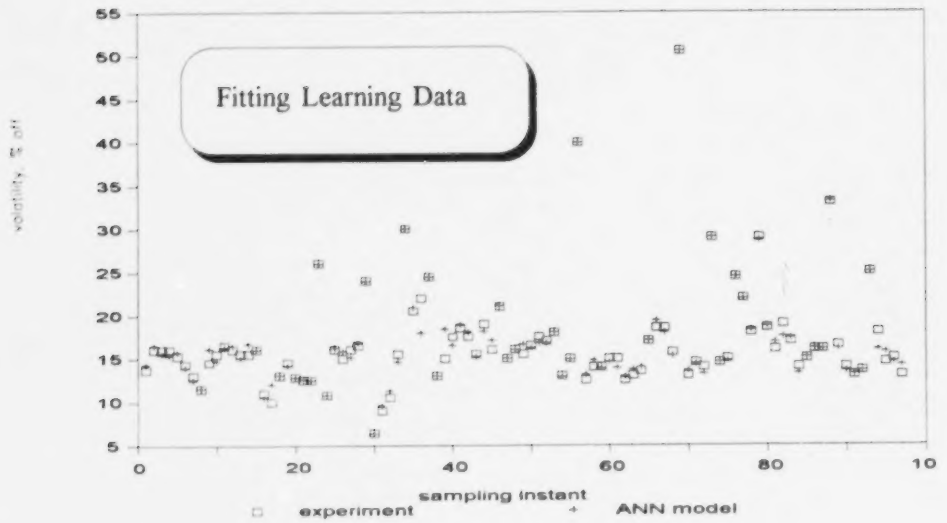


Figure 2.3.3.2(a) Fitting noisy plant stream data (Case Study II)

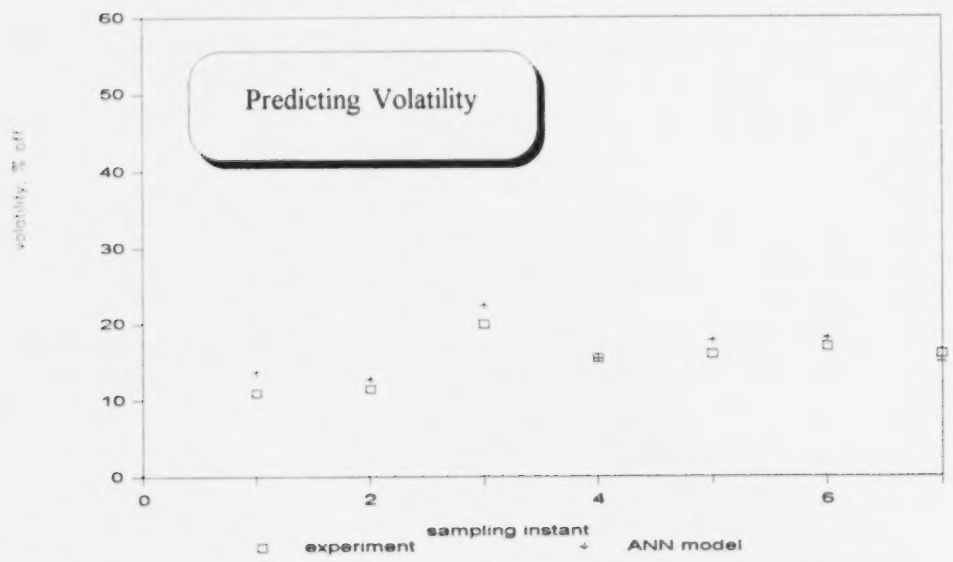


Figure 2.3.3.2(b) Prediction of volatility (Case Study II)

1. the longer training time): ANNs may not be a good choice for time-varying systems.

   Convergence and stability cannot be guaranteed.

2. Due to lack of robustness, some initial parameters(weights, learning rate,
3. etc.) may strongly influence the ANNs' performance: global assurance is needed.

Chapter 3

Integrating Neural Network to IDIS

This chapter first introduces fundamental concepts and methodology of Integrated Distributed Intelligent System (IDIS) as well as the meta-system environment, Meta-COOP, then discusses the integration of neural network, as a subsystem for numerical computational tool, into IDIS.

3.1 Integrated Distributed Intelligent Systems (IDIS)

As indicated in Chapter 1, the existing intelligent systems can only be used alone and inflexibly for a special purpose. It is difficult to integrate the isolated intelligent systems that have been available, even though each of them is well developed for a specific task. In general, the best way to solve this complicated

problem by intelligent system techniques is to distribute knowledge and to separate domain expertise. In such a case, several intelligent systems may be used together, each of them being developed to solve a particular sub-domain problem. However, conventional technology prohibits us from integrating several knowledge-based systems that have been successfully developed.

It is not difficult to find industrial problems that can not be solved by using the existing expert systems or numerical computation techniques. Many process variables that affect process operation and product quality and quantity are only partially understood, and sometimes may not be directly measurable. For example, some empirical data from industry cannot be effectively processed using existing expert systems or numerical computation packages alon. A new methodology integrating neural networks, symbolic reasoning systems, numerical computation packages as well as graphic simulation tools may be suitable to deal with such kind of data.

In the chemical industry, a vast amount of operation data and information from various sensors will be processed in time. So far, database technology and numerical computation methods have been used to deal with operation data. However, how to acquire knowledge effectively from the data source still remains a difficult issue.

Conflicts usually arise because different domain expert systems may make

different decisions on the same issues based on different criteria. Most intelligent systems for industrial processes work in an interdisciplinary field, and often deal with conflicting requirements. These intelligent systems may produce conflicting decisions based on even the same information. One of the bottlenecks for applying intelligent systems commercially is to integrate different quantitative and qualitative methods.

Distributed intelligent system technology is a very practical method for improving the ability of knowledge base management and maintenance (Bridgeland and Huhns, 1990; Conry et al., 1990). Since most industrial problems need knowledge and experience from different areas, the integration of distributed intelligent systems is a significant factor in solving more complicated industrial problems.

The modern industrial process is becoming increasingly difficult for operators to understand and operate effectively. Due to the importance of the operator's role in the systems, the quality of interaction between human operators and computers is crucial. Thus, developing an intelligent multimedia interface, which communicates with operators through multiple media and modes such as language, graphics, animation and video (Maybury, 1992), will facilitate human operators interaction with complex real-time monitoring and control systems.

Besides the above applications, the following drawbacks often arise in ex-

isting intelligent systems:

1. lack of efficient search methods to process different knowledge in a large decision-making space;
2. lack of coordination of symbolic reasoning, neural networks and numeric computation as well as graphics representation;
3. lack of integration of different intelligent systems, software packages and commercial AI tools;
4. lack of efficient management of intelligent systems;
5. lack of capability to handle conflicts among intelligent systems;
6. lack of a parallel configuration to deal with a multiplicity of knowledge representation and problem solving strategies;
7. difficulty in modifying knowledge bases by end-users rather than the original developers.

In solving the problems mentioned above, the key issue is how to develop a meta-system and implement multiple media integration. Concepts of integrated distributed intelligent system were proposed by Rao et al. (1987). So far, a prototype IDIS platform is available.

IDIS is a large knowledge integration environment, which consists of several symbolic reasoning systems, numerical computation programs, database management subsystem, computer graphics packages, multimedia interfaces, as well as a

meta-system. It makes use of the advanced object-oriented programming technique in C++ language. The integrated software environment allows the running of programs written in different languages, communication among subprograms, as well as the exchange of data between subprograms and databases. These isolated expert systems, numerical packages and programs are under the control of a supervising expert system, namely the meta-system. The meta-system manages the selection, coordination, operation and communication of these programs. The structure of the integrated distributed intelligent system is illustrated in Figure 3.1. An integrated distributed intelligent system has the following advantages:

1. provides an open architecture to help users protect their investment when introducing new computer technology;
2. coordinates all symbolic reasoning systems, numerical computation routines, neural networks, and computer graphics programs in an integrated environment;
3. distributes knowledge into separate expert systems, numeric routines and neural networks so that the knowledge bases of these systems are easier to change by commercial users other than their original developers;
4. acquires new knowledge efficiently;
5. finds a near optimal solution for conflicts and facts among different intelligent systems;
6. provides the capability of parallel processing in an integrated distributed intelligent system;

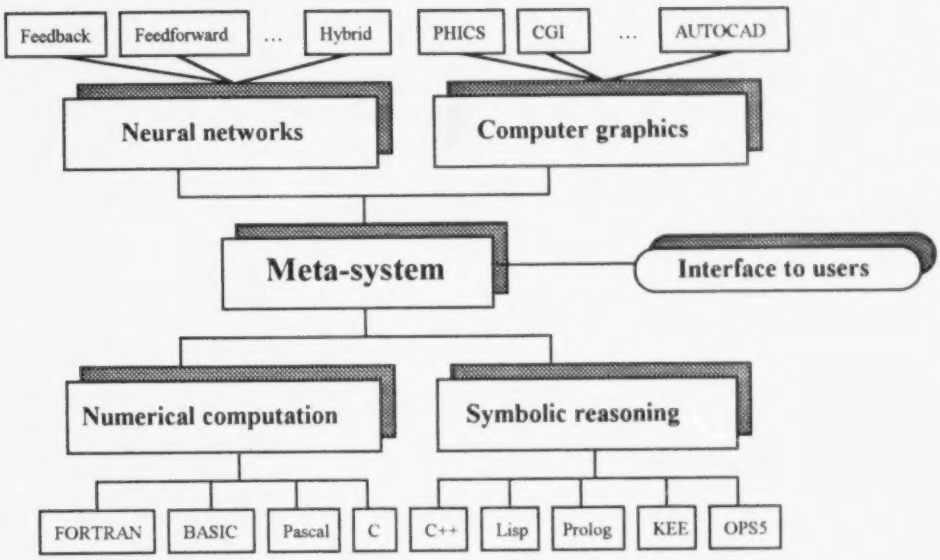


Figure 3.1 Integrated distributed intelligent system structure

7. communicates with measuring devices and final control elements in the control systems, and transforms various input/output signals into the standard communication formats.

IDIS is a key technology to the industrial automation at the knowledge-intensive stage. This new software integration platform can utilize different knowledge (analytical and heuristic knowledge), process different information (symbolic, numerical, and graphic information), and integrate different computer languages.

3.2 Meta-system

So far, many intelligent systems have been developed. However, their capability of dealing with complicated problems is very limited. The main disadvantages of the existing intelligent systems have been summarized in the previous section. The two most urgent problems to be solved are how to coordinate symbolic reasoning, numerical computation, neural networks, as well as computer graphics, and how to integrate heterogeneous intelligent systems.

It has been realized that integrating different intelligent systems into a large-scale knowledge environment is often necessary but difficult. In such an environment, a supervisory system that controls and manages the heterogeneous intelligent subsystems is required. The supervisory system has to provide integration functions in the following phases:

1. integration of knowledge of different disciplinary domains;
2. integration of empirical expertise and analytical knowledge;
3. integration of various objectives, such as research and development, system design and implementation, process operation and control;
4. integration of different symbolic processing systems (expert systems);
5. integration of different numerical computation packages;
6. integration of symbolic processing systems, numerical computation systems, neural networks, and computer graphics packages
7. integration of multimedia information, such as symbolic, numerical and graphic information.

Phases 1 and 2 are at the knowledge level. Phase 1 also indicates the characteristics of modern engineering techniques. Phase 3 functions at both the knowledge and the functional levels. Phases 4 through 7 perform their integration functions at the functional level, through the problem-solving level to the program level.

3.2.1 Functions

The key issue in constructing an integrated distributed intelligent system is to organize a meta-system. The meta-system can thus be referred to as a 'control mechanism of meta-level knowledge' (Rao, et al., 1987b). This new high-level supervisory intelligent system, that is, a meta-system, is available to control IDIS.

The main functions of a meta-system are described as follows:

1. Coordination function: The meta-system is the coordinator to manage all symbolic reasoning systems, numeric computation routines, neural networks, and computer graphics programs in an integrated distributed intelligent system.
2. Distribution function: The meta-system distributes knowledge to separate expert systems, numeric routines, neural networks, and computer graphics programs so that the integrated distributed intelligent system can be managed effectively. Such modularity makes the knowledge bases of these intelligent systems easier to change by commercial users rather than their original developers.
3. Integration function: The meta-system is the integrator which can help us easily acquire new knowledge.
4. Conflicting reasoning function: The meta-system can provide a near optimal solution for conflicting solutions and facts among different intelligent systems.
5. Parallel processing function: The meta-system provides the possibility of parallel processing in an integrated distributed intelligent system.
6. Communication function: The meta-system can communicate with the measuring devices and the final control elements in the control systems and transform various non-standard input/output signals into standard communication formats.

3.2.2 Configuration

Like the common expert systems, the meta-system has its own database, knowledge-base and inference engine, but it distributes its activities into the separated, strictly ordered phases of information gathering and processing. The meta-system configuration as shown in Figure 3.2, includes the following six main components: an interface to the external environment, an interface to internal subsystems, a meta-knowledge base, a global database, a static blackboard, and an inference mechanism.

3.2.2.1 Interface to External Environment

The interface to the external environment builds the communication between the users and internal software systems as well as among the external software systems. The interface plays a key role in an open structured software system in communicating with other intelligent software systems to extend the system into a much larger scale for more complicated tasks.

3.2.2.2 Meta-knowledge Base

The meta-knowledge base is the intelligence resource of the meta-system. It serves as the foundation for the meta-system to carry out the managerial tasks. The meta-knowledge base consists of a compiler and a structured frame knowl-

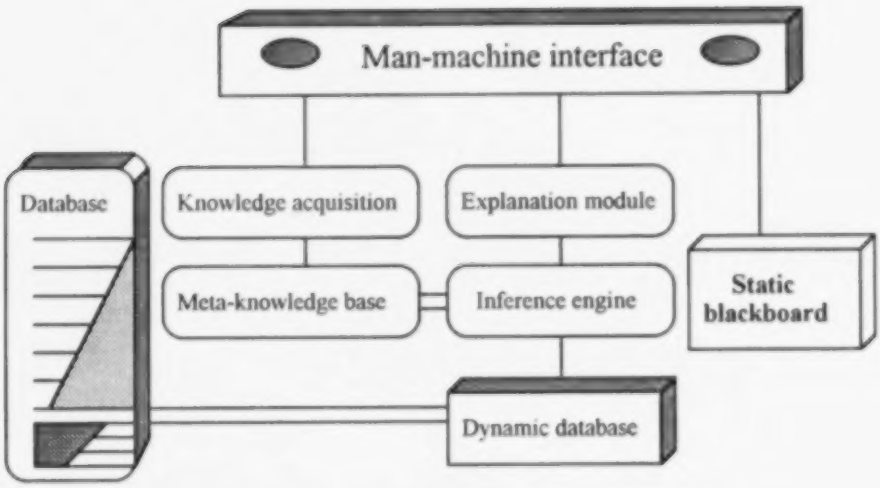


Figure 3.2 Meta-system configuration

edge representation facility. The compiler converts the external knowledge representation, which is obtained through the editor and is easy to understand by users, into internal representation forms available in the inference mechanism. The structure of knowledge representation can be production rule or frame or their combination. Characterized by diversity and variety in nature, the meta-knowledge may be better represented in object-oriented frame structures.

There are several modules in the frame to represent different components of meta-knowledge. These components are functioned for specific purposes. For example, the communication standardization module for heterogeneous subsystems and the conflicting resolution module are formed for a general management purpose at supervisory level. The module for knowledge about subsystems and the task assignment module have to be built according to each specific problem. The meta-knowledge base employs an open organization structure. It allows new intelligent functionality to enter the meta-knowledge base to engage more duties in decision-making.

3.2.2.3 Global Database

The database in the meta-system functions as a global database for the integrated distributed intelligent system, distinguished from the databases in the subsystems, which are attached only to their individual subsystems. The interface converts the external data representation form into an internal form. The data flow

in the global database is controlled either by the inference engine, depending on the corresponding module in the meta-knowledge base, or by users of certain security classes.

3.2.2.4 Inference Mechanism

Due to the diversity of the meta-knowledge and the variety of its representation forms, the inference mechanism in the meta-system adopts various inference methods, such as forward chaining, backward chaining, certain reasoning, uncertain reasoning, conflict reasoning, and so on. The inference mechanism performs operations and processing on the meta-knowledge. Additionally, it also carries out various actions based on the reasoning results; that is, passing data between any two subsystems, or storing new data in the database. Therefore, there are some functional modules in the mechanism, which further extends the functionality of the inference mechanism.

3.2.2.5 Static Blackboard

The static blackboard is an external memory for temporary storage of information that is needed when the system is running. Limited by the on-board memory space, the subsystems in IDIS are unable to execute at the same time. In fact, it is unnecessary to run the entire system simultaneously. Very often, the meta-system and all subsystems are run on the distributed hardware environment

so that there must be a buffer area in the external memory for any two subsystems to exchange information. Besides data storage, the conversion of data in heterogeneous languages into exchangeable standard form is also completed in the static blackboard.

3.2.2.6 Interface to Internal Subsystems

This component of the meta-system is established based on each specific application. The internal interface connects any individual subsystems which are used in problem-solving and under the control and management of the meta-system. Each module of the interface converts a nonstandard data form from a specific subsystem into a standard form in the integrated distributed intelligence environment. Conversion among the standard forms of different languages is carried out by the meta-system.

3.3 Meta-COOP

To successfully implement the functions of the meta-system, a new structure of the meta-system, namely Meta-COOP, is currently available. Meta-COOP coded in C++ utilizes the objective-oriented programming technique and runs under UNIX™, VMS™ and DOS™ operating systems for the SUN workstation, VAX and PC 486 machines. Meta-COOP provides such distinct characteristics as the integration of various knowledge representations and inference methods.

The process to solve a complicated problem is actually a synthesizing process, which employs various knowledges and problem-solving strategies. In Meta-COOP, the organization structure of the knowledge-base can be divided into several components. Meta-COOP adopts the object-oriented programming technique and frame-based knowledge representation to implement the organization, management, maintenance and applications as a complex knowledge base system.

Meta-COOP distributes its meta-knowledge into many knowledge bases. Each knowledge base can be viewed as a fundamental knowledge unit (this may be a set of rules, a set of operation commands, or numerical models) to deal with an industrial problem and attached to a frame or an object that represents the problem. Frame-based or object-based knowledge representation not only describes in detail the attributes of an object but also hierarchically constructs the general knowledge base system, thus expressing the internal relationship among the knowledge bases. Such a hierarchy often has the two structures: classification structure and decomposition structure.

The decomposition structure stands for the organizational characteristics of an object in nature. Each part to be decomposed is always an organic part of the whole object. If some parts in the decomposition structure are ignored, the whole system may lose its original characteristics. Such a decomposition structure can hierarchically represent the organizational construction and interrelations of knowledge bases.

The procedure to solve a complicated industrial problem consists of two stages: the first stage is to decompose a complicated problem into many basic sub-problems (from general to particular or from top to bottom). At the second stage, the sub-problems are solved individually. Meta-COOP provides the intelligent facilities for implementing this procedure. It resolves a complicated problem using a frame-based hierarchical structure and accomplishes communication and conflicting coordination between sub-problems through message sending. Each sub-problem or subsystem (it is always viewed as an object in Meta-COOP) and the interrelations can be described by the frame-based structure, while a knowledge unit to handle the sub-problem is closely attached to a slot in a frame. A knowledge unit may be constructed by a set of production rules, a set of operation commands, analysis methods, external procedures written in any other languages, and internal subroutines written in C™.

The classification structure describes the characteristics selection of a subsystem or sub-problem. The classification relationship can be expressed by a Superclass-Class-Subclass hierarchical structure in Meta-COOP.

With Object Oriented Programming (OOP) technique, users can effectively design a hierarchical knowledge structure that makes the knowledge base systematized, modularized, and easier to understand, maintain and manage. Using the decomposition and classification structures, the meta-system can be expressed as a complicated tree structure available to represent a real problem.

Meta-COOP supports a variety of knowledge representations such as frame-based, rule-based and method-based representations. Different from other expert systems, the communication among the representations is the most important characteristic. Since Meta-COOP uses various knowledge representations and problem-solving strategies, its inference mechanisms and information exchange in problem-solving are more complicated than those in conventional intelligent systems. Figure 3.3 demonstrates the fundamental control structure.

1. Method base: A method base is a specific procedural knowledge that is attached to the frame. It consists of two parts: an information filter (a group of keywords) and a method body. The information filter determines if a method can be triggered or called when the sending message goes through the filter. The method body can perform symbolic reasoning and numerical computation, control the problem-solving process, and send a message to other frames.
2. Frame-rule base: A frame is a fundamental unit in the meta-knowledge base. It expresses the natural attributes of an object with slots and facets. It also clearly represents the topological relationship among objects with the decomposition and classification structures. A set of rules as a slot value is stored in a slot of the current frame. This slot is named as the rule slot.
3. Database: The meta-system has a miniature relational database to record the intermediate results.
4. Method-based inference: The inference engine is used to perform a method

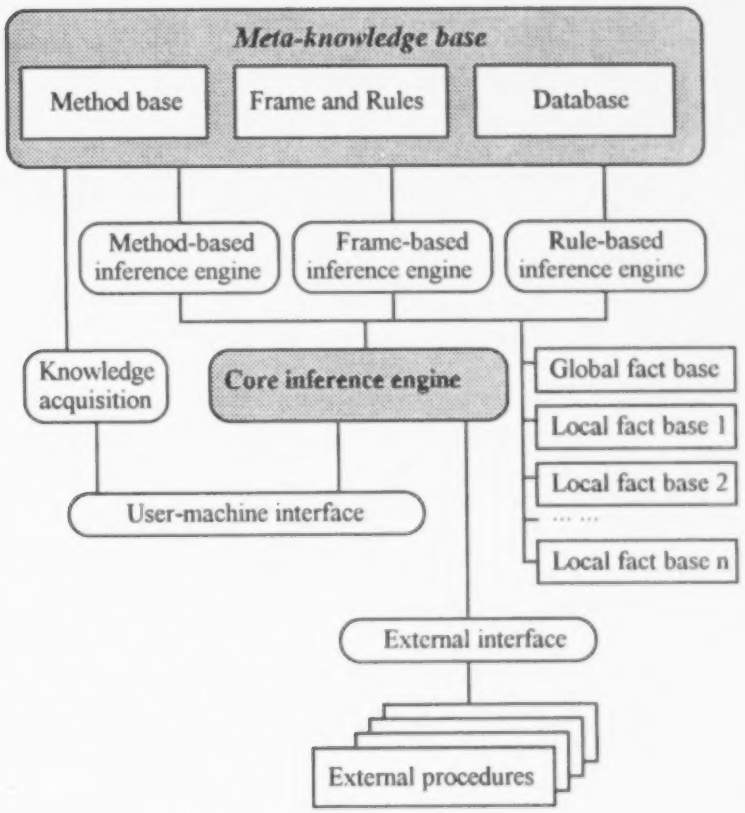


Figure 3.3 Integration of inference strategies in Meta-COOP

body, to implement numerical computation, and to select search paths. In addition, it can also invoke complex external procedures.

5. Frame-based inference: This performs an evaluation manipulation on the current frame, and inherits information from other frames. The frame-based inference accomplishes data processing, and implements the evaluation on a method or a set of rules, that is, it can find the specified rules set or methods, then call the corresponding method-based inference or rule-based inference to process.
6. Rule-based inference: Meta-COOP includes the production rule reasoning mechanism with a forward chaining control strategy to manipulate the rule-based knowledge unit attached to a rule slot in the current frame. It can load a conclusion through inference into any slot in other frames, but cannot trigger the frames. The task to activate other frames is performed by the method-based inference.
7. Core inference: The core inference engine can invoke the low-level inference engines and coordinate the conflicts among them in terms of the operational requirements. It also performs the communication between subtasks and message sending.
8. Global fact base: The global fact base stores the intermediate facts and results shared by all inference engines. These facts are often generated by the rule-based inference engine and called by other inference engines.
9. Local fact bases: In order to satisfy the software engineering requirements (information hiding and modularized knowledge base), Meta-COOP uses the

local fact bases to store the local facts that are hidden in a frame or an object. These local facts are derived from the method-based inference and only used in evaluating methods.

10. External interface: Meta-COOP is an open system to help users extend software functions. The external interface allows users to define some functions or procedures written in C as the internal functions of Meta-COOP. The functions are called by the method-based inference engine. The external interface also sets up the communication facility for message sending and result returning.
11. Knowledge acquisition: The module facilitates editing and compiling a knowledge base. The source codes of the knowledge base can be generated by common editors such as VI™, PE2™, etc., then stored as ASCII files. After the files are compiled, the ASCII file is transferred into the binary file that is directly referred to the inference engines.
12. Man-machine interface: This receives the input information from users and provides the problem-solving results and interpretation of reasoning processes.

3.4 Integration of Neural Network into IDIS

As illustrated in Chapter 2, ANNs can be often used to predict an output on the basis of several inputs. In the application of ANN to the inferential modelling

for the prediction of the desired product volaltility in the Sarnia Refinery (Case Study I), a comparison of the performance has been made between the RA model and the ANN model in predicting the desired product volatility with process stream data. However, the relative importance of the input variables in the inferential model have not been explained. Due to difficulties in explaining how ANN can reach its result, it is normally regarded as a kind of black box. It has been pointed out that physical meanings about the process modelled by ANN can not be extracted from the training results containing connection weight values.

It is obvious that every input variable can be dropped out from the model only if its associated connection weights between the input and output layers are all zeros. Even when those weights approach zero, removing that input variable from the model can be also considered because it may have very little impact on the output predictions. Hinton's diagram analysis a good example. In real situations, however, it really never happens.

An alternative approach for the evaluation of ANN inferences is using the connection weights between layers to partition the relative share of the output prediction associated with each input variable (Garson, 1991). Then the partition can be used to make judgments about the relative importance of input variables in a model. For the development of an integrated process modelling environment, the connection weight interpretation tool has been also integrated into IDIS along with the neural network.

3.4.1 Interpreting Connection Weights

ANN can be trained to model the process with the provided learning data set (input and desired output) by the chosen learning paradigm. The connections and their weights in an ANN can be regarded as the knowledgebase in an expert system. It is essential to analyze its content, in order to understand the relative causal importance and order of the input variables, by the use of the input-layer connection weight partition (Garson, 1991).

In partitioning connection weights between the hidden and output layers into input node shares, it is necessary to have an approach that focuses on the output rather than input-layer connection weight. The value of the hidden nodes is a function of the weights of the paths from the input nodes, and the output node's value is a function of the weights of the paths from the hidden nodes. The weight along the paths from the input to the output node indicate the relative predictive importance of the independents. These weights can be used to partition the sum of effects on the output layer, as followed, while using absolute values of all weights.

$$\text{\% contribution of ith input on jth output} = \frac{\sum_{k=1}^{N_1}\left(\frac{|W_{i,k,1}|}{\sum_{l=1}^{N_0}|W_{l,k,1}|}|W_{k,j,2}|\right)}{\sum_{m=1}^{N_0}\sum_{k=1}^{N_1}\left(\frac{|W_{i,k,1}|}{\sum_{l=1}^{N_0}|W_{l,k,1}|}|W_{k,j,2}|\right)} \qquad (3.4.1.1)$$

Since ANN doesn't incorporate this procedure, Connection Weight Interpretor (CWI), a tool to evaluate the percent contribution of input variables in the ANN model on the basis of connection weights, has been independently deveoped in this study. Then, it has been also integrated into IDIS via Meta-COOP. This integrated process modelling approach using ANN and CWI can provide a systematic way of using neural networking for modeling as well as for prediction.

3.4.2 Implementation

IDIS faces two barricades for integrating the neural network and connection weight interpretator on a single computer. First, due to the limitation of the on-board memory space, the whole system may not be able to be loaded at the same time. Secondly, due to the heterogeneity of the subsystems developed in different languages, the system may not be linked together as a whole and loaded in at the same time. Therefore, a technique, namely covered-structure, can be adopted to solve these problems. The basic strategy of the technique is to invoke the subsystems in turn by an order, controlled by the meta-system, in accordance with the memory space and the nature of the subsystems, then find some way to allow information exchange between the invoked subsystem and the ones at rest.

Shown in Figure 3.4.2.1, there are two ways for communication, namely, direct communication and indirect communication. The direct communication method is preferred for those subprograms which are used frequently and require

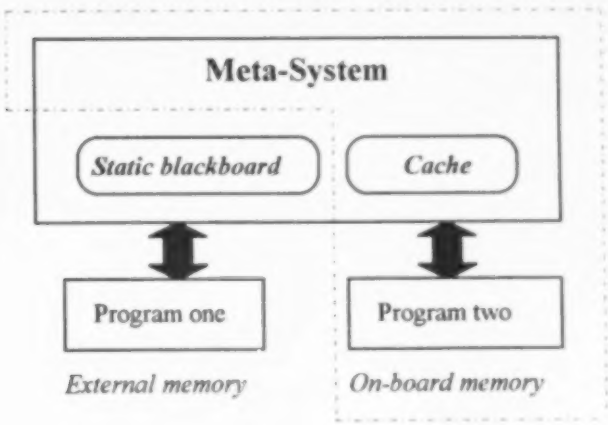


Figure 3.4.2.1 Communication between meta-system and subsystems

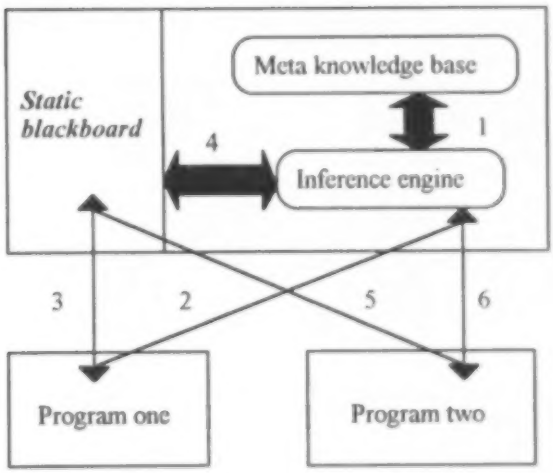


Figure 3.4.2.2 Indirect communication procedure

quick access to the meta-system in order to transform data quickly. Under the conditions of enough memory and compatibility of the languages, the subprograms may be compiled and linked with the meta-system and reside in the memory for the whole process. In this way, the meta-system can communicate with the subprograms directly.

Indirect communication is realized through a 'transformation station', namely a static blackboard. The meta-system can invoke some executable subprograms by using system call. The results from the running subprograms are then sent to the static blackboard, and processed there into the standard forms for calling by other subsystems through the meta-system such a manner. The procedure of indirect communication is illustrated in Figure 3.4.2.2 and takes several steps, as follows:

1. Based on the meta-knowledge, the inference engine sends program P1 the instruction. Program P2 obtains data from the instruction, then passes the language type of P2 to the instruction.
2. The inference engine invokes P1, and P1 starts to run.
3. The running results of P1 are written in SB with a file name, such as 1.DAT. The data file informs the meta-system.
4. The meta-system identifies the data file by the name 1.DAT and standardizes it, then saves the results in S1.DAT.
5. The meta-system sends S1.DAT to program P2.

For the integration of neural network as well as connection weight interpretation tool into IDIS, Meta-COOP has been utilized to develop the indirect communication interface that allows each subprogram swap data interactively. The data flow diagram for neural network, connection weight interpretator and expert system under the supervision of meta-system is shown in Figure 3.4.2.3. The knowledge base source code is shown in Appendix B. Rules 1 and 2 check whether the user want to train the neural network or to predict the product quality. Based on the user response, the meta-system transmits the appropriate information to the neural network. Once the neural network has completed training or predicting the product quality, the meta-system extracts the required data from the neural network output. If training is the case, the meta-system send the training result to the connection weight interpretator. Then, the interpretator partition the connection weight, as explained in the previous section, to find the significant variable contributions in the model. The meta-system takes the percent contribution of each independent variable and suggests which variable can be dropped out. If prediction is the case, the meta-system sends the predicted product quality to any other subsystem, for example, other expert systems for intelligent process operation.

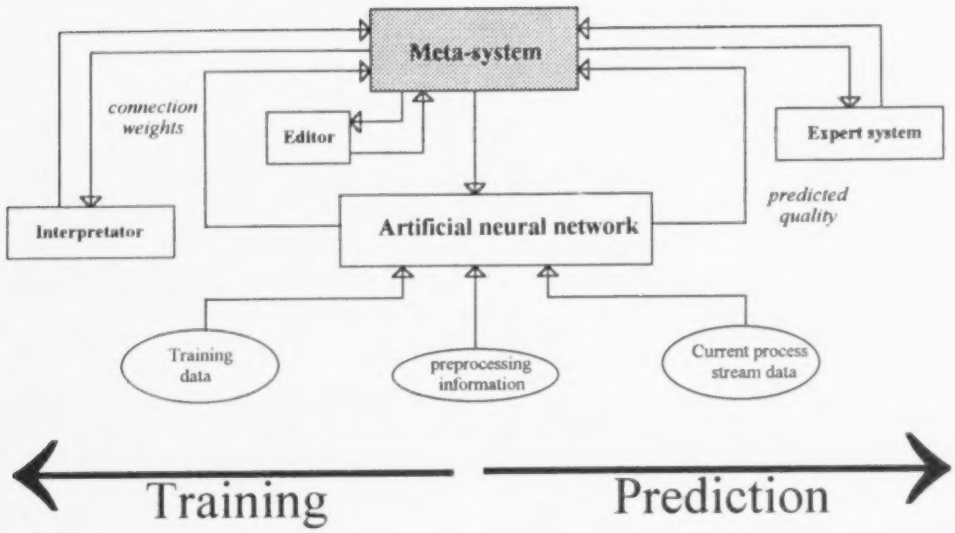


Figure 3.4.2.3 Data flow diagram for ANN, CWI and any other subsystem under the supervision of a meta-system

Chapter 4

Integrated Process Control System Design*

This chapter describes the integration strategy to develop an interactive Computer-Aided Control System Design (CACSD) environment for chemical processes. An interactive graphics software package PCET (Process Control Engineering Teachware) developed in this study is also presented. PCET has a hierarchical structure of several independently developed subprograms which are integrated under the control of a supervising system, meta-system. It covers a wide spectrum of process control engineering applications, including Time Domain Analysis (TDA), Routh Stability Criterion (RSC), Root Locus Technique (RLT),

* A version of this chapter has been presented at the 1992 IEEE Symposium on Computer-Aided Control System Design, March 17-19, 1992 in Napa, California.

Frequency Domain Analysis (FDA), Linear State Space Analysis (LSA), Discrete -time System Analysis (DSA) and Industrial Application Case (IAC).

4.1 Introduction

The design of control system is by its very nature an iterative process. Usually, several trials have to be made until a reasonable control system design is found. The design by pencil and paper requires much work, for examples drawing frequency responses and root locus, calculating dominant pole response, etc. However, the results obtained are only approximate ones. In many cases, the exact analytical solutions are not even available. Such an example is the closed-loop response of a linear system with dead time. In fact, the design can be considerably improved by the use of computers. Methods that are not important for the understanding of the subject matter, such as the manual drawing of root locus curves, can be removed, giving users the time to address additional control concepts. Moreover, many problems arising in control theory are solvable only by numerical methods. Thereby, the essential facets of control engineering can be brought to bear over a shorter time-span.

During the last decade, with the advent of microcomputer technology, there has been a rapid growth of interest in the use of computers for the design of control systems. It has been widely recognized that a comprehensive CACSD

software must have the following essential characteristics:

1. interactivity
2. modularity and flexibility
3. graphical capabilities
4. reliable computational methods
5. broad spectrum of control engineering
6. integrated environment

A large number of CACSD packages have been already developed by universities and industry. However, most existing CACSD packages are often specific and inflexible such that they deal with particular aspects of the overall control system design problem. These problems can be solved by utilizing a meta-system concept in the integration of several programs, dealing with the different stages in the design process. The meta-system concept was proposed by Rao et al. in 1987, which was developed to handle the integration issue for distributed intelligent systems. In this study, the meta-system concept is applied to conventional control system design. Current CACSD packages are also difficult to use effectively, and unhandy in their man-machine interface. Since these packages are mostly in command-driven environments, their users are expected to be familiar with many commands, and to know their correct sequence in the right syntax. One of the main advantages of CACSD systems is that the sophisticated methods can be more easily understood, accepted and applied by users. For this reason, the

program packages must be easy to learn and to use. Based on this point of view, the CACSD package PCET, developed in this study with the BASIC programming language for IBM PC's and compatibles, is programmed as a very user-friendly software environment. Hence, its users can run the program without the knowledge of the contents of the package, commands and the language.

4.2 System Construction

The following sections present the integration strategy, the implementation philosophy and the integrated control system design environment. Overall system structure of PCET and its functions are briefly described.

4.2.1 Integration and Management of CACSD

As mentioned previously, current CACSD packages normally suffer from the limitation of their modularity and flexibility since they are often specific and inflexible such that they deal with particular aspects of the overall control system design problem without unifying them in a comprehensive design environment. Thus there is a need to develop a more general CACSD software which would integrate a wide spectrum of process control engineering within a powerful conceptual framework. The development of such an integrated CACSD environment can be

approached by combining several existing control packages into a comprehensive design environment. The main disadvantages of this approach is that the end users have to know several different systems. Especially, when they use the command language communication method as a man-machine interaction interface, the number of commands, which the end users have to remember, become enormous. And the users may often confuse which command belongs to which system. In order to solve these problems, an integration strategy to coordinate several independently developed subprograms and to deal with the different stages in the design process, is implemented in PCET. These independently developed subprograms are integrated under the control of a supervisory system, meta-system. This approach originated from the integrated distributed intelligent system concept.

Even though the meta-system concept was originally developed for the integrated distributed intelligent system (i.e. the integration of expert systems and numerical computation packages), this study aims at extending the meta-system concept application to conventional control system design. When using meta-system concept in CACSD environment, individual subprograms can be developed and applied for the specific purpose at different times. Such a configuration allows us to write, debug and modify each program separately so that the overall integrated design environment can be managed efficiently. This approach makes the integrated design environment much easier to be modified. The meta-system also provides us a free hand in integrating and utilizing new subprograms. Any communication between two programs must rely on the translation of the meta-system.

Such a configuration enables us to add or delete subprograms easily. When a new subprogram is to be integrated into the integrated CACSD environment, we just have to modify the interface of meta-system, while other subprograms are remain unchanged.

4.2.2 Integrated Control System Design Environment

In this section, advantages of the integrated control system design environment developed in this study are discussed. The overall structure of the interactive control system design software package PCET is also described.

4.2.2.1 User-friendly Interface

When designing an interactive system, the first step is to decide the form of the man-machine interaction to take place. Although the man-machine interface is not the main part of any CACSD system, this decision is very important because the mode of interaction determines the user-friendliness and the user acceptance of the system.

There are four basically different ways for interactive input commonly used: question-and-answer method, menu-driven operation, command-language communication, and graphical input. The first three methods with alphanumeric informa-

tion at least in principle can work on any alphanumerical terminal, whereas the graphical input requires special hardware in form of a graphical terminal for the graphic echo and some graphical input device. For CACSD packages primarily designed to solve numerical problems, the first three alphanumeric input modes are of interest. Of these first three, the command language communication mode lets the user be in charge of the conversation. With command-language communication interaction, the user is expected to be familiar with over a hundred commands, and to know the correct sequence of them in the right syntax. For example, the user must know how to form a transfer function, and how to specify a variable as parameter for a function, and so on. Even this low level of complexity can be still frustrating to the non-expert users. Whereas, the menu-driven and the question-and-answer methods, which are somewhat less flexible but very user-friendly, give the computer almost total control. Menu-driven interactive programs are extremely handy and speedy for dexterous users, and the question-and-answer method may be the best choice for minimal learning time and maximum accessibility by non-experts.

For a maximum efficiency in CACSD environment, PCET uses the combinations of the question-and-answer and the menu-driven interaction methods so that it provides a manual-free operation environment. Thus, it requires neither commands nor a manual to use PCET. With this very user-friendly interactive interface, specially advantageous for users who are not familiar with the system, the application can be learned in a few hours without special knowledge of the

detailed contents of the software and the language.

A user-friendly environment also appears in model selection. A polynomial model or a pole-zero model can be chosen to represent the transfer function of a process. It is convenient for users to use the different models to analyze a system or to understand concepts. For instance, pole-zero model is more suitable to understand the concept of root locus, while a polynomial model is more appropriate for studying Routh stability criterion.

4.2.2.2 Graphic Display Capability

The use of graphical outputs (step response, root locus, Bode diagram, etc.) has become an indispensable part of many modern CACSD packages. In the analysis and design of control system, diagrams or graphs generated by PCET can give the users an intuitive feeling for the quantitative and qualitative aspects of system behavior. When this software is used for education, students can gain a clear insight into theoretical concepts and practical computer simulation experience.

The required graphical facilities, together with the needed computational power, dictate the hardware requirements for CACSD applications. Modern workstations, which may be an ideal environment for the dedicated control scientists, provide excellent graphics as well as enough computing power. However,

the price per computer-connection is rather high for the workstations and comparable to that multi-user minicomputer environment which is another very common configuration in CACSD application. Since in the educational process hundreds of students have to use computers for their exercises, more economical solutions have to be found so that students can access the software package easily. Thanks to the rapid development of microelectronics technology, modern personal computers can provide the user with decent graphical capabilities and enough computing power for introductory to intermediate control exercises. Moreover, packages usable in control exercises have already emerged for the PC's (e.g. PC-MATLAB, Program CC, TUTSIM, etc.). For these reasons, PCET was developed for modern personal computer usage (e.g. IBM-PC's or compatibles). It is given on a diskette and can be distributed to users in order to solve many practical process control problems (Rao and Qiu, 1993).

4.2.2.3 Integrated Distributed System Configuration

As discussed in Section 4.2.1, independently developed subprograms are integrated under the control of a supervisory system, meta-system. This approach originated from the integrated distributed intelligent system (IDIS) concept, which was proposed by Rao et al. in 1987 in order to handle the integration issue for distributed intelligent systems.

The IDIS is a large knowledge integration environment, which consists of

several symbolic reasoning systems and numerical computation packages. These software programs may be written in different languages and be used independently. They are under the control of a supervising intelligent system, meta-system. The meta-system manages the selection, operation and communication of these programs. By integrating different intelligent systems, the efficiency as well as conceptual and structural advantages can be achieved (Rao et al., 1987). This new conceptual design framework can serve as a universal configuration to develop high-performance intelligent systems for complicated applications. The key issue to construct the IDIS is to organize a meta-system. The meta-system controls the selection and application of all programs in IDIS and executes the translation between different data or programs.

Even though the meta-system concept was originally developed for the IDIS, it can be still applied to the integration issue of CACSD packages. By integrating several independently developed subprograms under the control of a supervising system, meta-system, PCET has a hierarchical structure as shown in Figure 4.2.1. In PCET, the meta-system serves as an interface mechanism to control and communicate individual subprograms developed for different purposes, and to build a man-machine interface. The advantage of this approach for the development of CACSD environment is that any other subprogram can be added to the system very easily. When a new subprogram is to be integrated into PCET, only the interface of the meta-system need to be modified without interrupting other subprograms. It also allows each subprogram to be modified without affecting

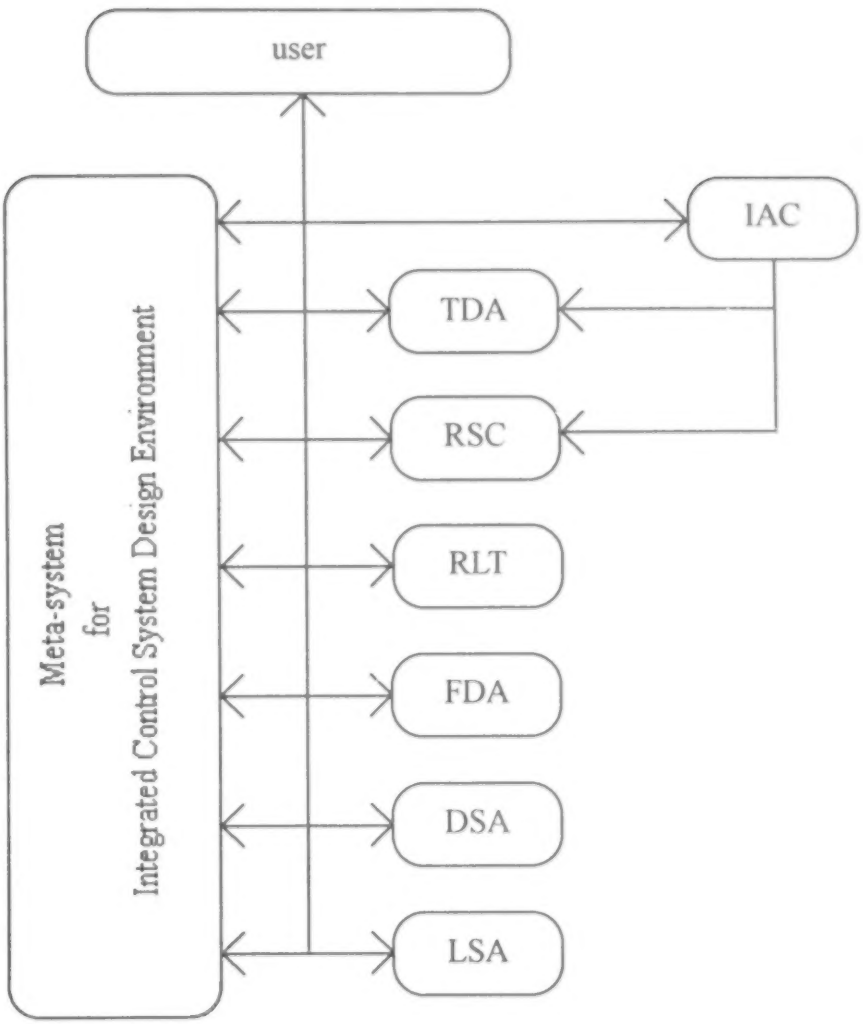


Figure 4.2.1 Overall hierarchical structure of PCET

others, and to be executed separately as a stand-alone system.

A global data base is set up in the meta-system, which is a common data resource for subprograms. The process model can be put in at any subprogram and shared with other subprograms. With this configuration, only one-time data input is necessary to analyze the same model in different domains or with different methods. The model can also be saved for later use.

4.2.2.4 Industrial Process Control Application Case

Chemical processes are very complicated, with a high concentration of advanced technology and operation automation. It is important for engineering students to have basic practical experience before they start their professional careers. Industry welcomes students who have more practical experience. Unfortunately, no such application cases have been implemented in an education-oriented control system software.

Process control is an application-oriented area. When real industrial processes or appropriate laboratory equipments are not accessible, computer simulation can give the users the practical knowledge about industrial applications. For example, pulp and paper industry is one of the most important industry in Canada. Pulp and paper production is a very complicated process. However university students do not have any idea about the process and the control system. Intro-

ducing a pulp and paper process in PCET can benefit both the pulp and paper industry and university.

A paper machine headbox is chosen as a typical industrial case study in order to combine theory with practice, and to link abstract mathematical models with the real industrial process. A paper machine headbox control system is widely used in pulp and paper mills to control basis weight and moisture of the products. The main problems encountered with headbox control are the ability to maintain the total pressure and level in the headbox, and to maintain the rush/drag ratio when wire speed changes. Different headbox control schemes should be chosen according to the different headbox types and stock delivery systems. With a model of the paper machine headbox control system, several typical industrial process control configurations, such as single-loop control and cascade control, are also incorporated in PCET. Under every control configuration, several subprograms are linked. They currently includes Time Domain Analysis (TDA) and Routh Stability Criterion (RSC). The introduction to the process and the schematic flow diagrams of the control systems are included in PCET.

4.3 Functions of PCET

As shown in Figure 4.3.1 which is a hardcopy of program menu screen, PCET covers a wide spectrum of process control engineering applications, including Time Domain Analysis (TDA), Routh Stability Criterion (RSC), Root Locus

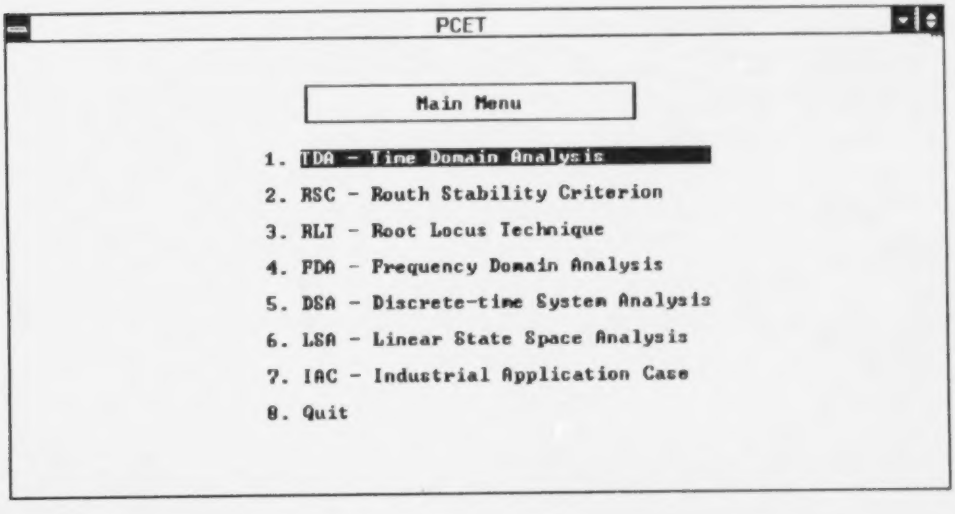

PCET
Main Menu
1. TDA - Time Domain Analysis
2. RSC - Routh Stability Criterion
3. RLT - Root Locus Technique
4. FDA - Frequency Domain Analysis
5. DSA - Discrete-time System Analysis
6. LSA - Linear State Space Analysis
7. IAC - Industrial Application Case
8. Quit

Figure 4.3.1 Main menu of PCET

Technique (RLT), Frequency Domain Analysis (FDA), Discrete-time System Analysis (DSA), Linear State Space Analysis (LSA), and paper machine headbox control as a Industrial Application Case (IAC). Besides design applications, this software can be also used to improve understanding of the basics of process control engineering, and helps users or students to gain practical experience on simulation and computer-aided design.

Functions of PCET are summarized in Table 4.3.1 and the details of system functions will be discussed in Chapter 5. For example, TDA includes the open-loop and closed loop transient responses for a linear, time invariant system with or without time delay, analysis of control algorithms and controller action, parameter tuning of PID controllers. In this program, students can learn not only fundamental knowledge such as open-loop, closed-loop and feedback, but also additional important concepts such as process gain, time constant and dead time. Time delay is an important characteristic in chemical processes, as it is often encountered in many industrial processes, such as reactors and heat exchangers. Most control laws applied to the present industrial environment are PID control algorithms. Therefore, the analysis of PID control algorithms and PID controller's action are incorporated in TDA.

4.4 System Evaluation

PCET, developed in this study, has been used in the process control courses,

Acronym	Explanation	Functions	Applications
TDA	Time Domain Analysis	Open-loop system response Clsosed-loop system response PID control algorithms	Controller analysis System dynamic analysis System design Controller tuning
RSC	Routh Stability Criterion	Stability testing Routh array computation Pole calculation	Stability analysis
RLT	Root Locus Technique	Root locus plots Characteristic roots Critical points	System dynamic analysis Stability analysis Pole assignment
FDA	Frequency Domain Analysis	Nyquist plots Bode diagrams Gain and phase margin	System dynamic analysis Stability analysis Controller design
DSA	Discrete-time System Analysis	Open-loop system response Clsosed-loop system response PID control algorithms	Controller analysis System dynamic analysis System design Controller tuning
LSA	Linear State Space Analysis	Controllability matrix Observability matrix Eigenvalue and eigenvector Matrix manipulation	Controllability analysis Observability analysis Eigenstructure analysis
IAC	Industrial Application Case	Pulp and paper machine	Single loop control Cascade control

Table 4.3.1 PCET functions

ChE 546 'Process Dynamics and Control' (Springs of 1991, 1992 and 1993) for senior undergraduate students and ChE 646 'Process Dynamics and Computer Process Control' (Falls of 1991 and 1992) for graduate students, offered by the Department of Chemical Engineering at the University of Alberta. Using PCET, students directly perceived the theoretical concepts and gained practical experience. At the end of courses, the PCET evaluation forms (Appendix C) were distributed to students for the improvement on this software package.

This software also attracted the industry's attention. Several companies provided financial supports and their knowledge for the development of this software package. It has been used in the both university and industrial plants. The old version of PCET was demonstrated in the International Federation of Automatic Control (IFAC) Conference on Advances in Control Education held at Boston in June 1991.

4.5 Conclusions

The integration strategy to design an interactive computer-aided control system design environment for chemical processes is presented, in which several independently developed subprograms are coordinated by a meta-system. By this kind of integration approach, each subprogram can be easily modified without affecting other subprograms. It also allows any other subprograms to be added

to the system easily. Moreover, individual subprograms can be executed separately as a stand-alone system. The interactive graphics software package, PCET, has a hierarchical structure of several independently developed subprograms which are integrated by a meta-system. In the software package PCET, the meta-system serves as an interface mechanism to communicate individual software programs, and to build man-machine interface. Currently, PCET, written in the BASIC programming language for IBM PC's and compatibles, covers a wide spectrum of process control engineering, including Time Domain Analysis (TDA), Routh Stability Criteria (RSC), Root Locus Technique (RLT), Frequency Domain Analysis (FDA), Discrete System Analysis (DSA), Linear State Space Analysis (LSA) and Industrial Application Case (IAC). This software can be used to design control systems. It can also improve understanding of the basics of process control engineering, and helps users to gain practical experience on simulation and computer-aided design.

Chapter 5

Process Control Engineering Teachware

This chapter briefly describes the basic concepts and theoretical fundamentals of process control systems, on which the interactive software package PCET is based. Illustrative examples solved by using the subprograms in PCET are also included in the respective sections.

5.1 Time Domain Analysis (TDA)

5.1.1 Step Input

When designing process control systems, the engineer requires an objective

means of evaluating the performance of the control schemes under consideration. Performance criteria are quantitative measures of the system response to a specific input or test signals. In the subprogram TDA, the step function is used in evaluating the performance of control systems for the following reasons:

1. It is simple and easy to produce.
2. It is by far the most common form of test signal used in practice.
3. It is considered to be a serious disturbance, since it is capable of changing the mean level of the process. Many other kinds of bounded disturbances can be overcome if a step disturbance can be overcome.
4. The response of the system to other types of disturbances can be inferred from the process step response.
5. The step response is easy to measure, and thus get an approximated transfer function of the system.

A step input is a sudden and sustained change in input defined mathematically as

$$u(t) = \begin{cases} M & t \geq 0 \\ 0 & t < 0 \end{cases} \qquad (5.1.1.1)$$

The constant M is known as the magnitude or size of the step function. u(t) is called a unit step function if M=1. The Laplace transform of a step function is

$$U(s) = \frac{M}{s} \qquad (5.1.1.2)$$

The response of a system to a step input is referred to as its step response, and conveys information regarding the dynamic and steady state behavior of the system.

5.1.2 Transfer Function

A transfer function is an s-domain algebraic representation of the relationship between the input and output variables of a system. It is typically obtained by taking Laplace transform of a system differential equation with zero initial conditions. A transfer function completely determines both the dynamic and steady state behaviors of the process output when the input signal is specified.

For physically realizable systems, the degree of the denominator polynomial of the corresponding transfer function always exceeds that of the numerator. If the order of the numerator is larger than that of the denominator, the transfer function is said to be unrealizable.

The roots of the numerator polynomials are called the zeros of the transfer function. If s assumes the value of any zero, the transfer function vanishes. The roots of the denominator are known as the poles of the transfer function. When s approaches a pole, the transfer function becomes infinite. The poles and zeros of the process transfer function play an important role in the analysis and design of control systems.

The PCET software package developed in this research enables the user to examine the dynamics of up to 5th order systems using the subprogram TDA. The process/system models can be in three types as follows:

1. Model 1 ($n \leq 5$, $m \leq 4$): A polynomial model:

$$G(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_1 s + a_0}{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0} e^{-ts} \qquad (5.1.2.1)$$

2. Model 2 ($n \leq 5$, $m \leq 4$): A zero-pole model:

$$G(s) = K_0 \frac{(s - z_m)(s - z_{m-1}) \cdots (s - z_0)}{(s - p_n)(s - p_{n-1}) \cdots (s - p_0)} e^{-ts} \qquad (5.1.2.2)$$

3. Model 3: A typical process model:

$$G(s) = \frac{K_0}{(T_1 s + 1)(T_1^2 s^2 + 2T_1 x s + 1)} e^{-ts} \qquad (5.1.2.3)$$

5.1.3 PID Controller

Most control laws applied to the present industrial environment are PID control algorithms. Therefore, the analysis of PID control algorithms and PID controller's action are incorporated in TDA. A PID controller includes three modes: proportional, integral and derivative actions. Therefor, the PID controller

equation is formed by summing the contributions of the three basic control modes, and its Laplace transform is

$$G_C(s) = K_C\left(1 + \frac{1}{t_I s} + t_D s\right) \qquad (5.1.3.1)$$

The response of a PID controller to a unit step change in actuating error can be simulated with the subprogram TDA in PCET.

5.1.4 Time Delay

Time delay is an important characteristic in chemical processes, as it is often encountered in many industrial processes, such as reactors and heat exchangers, where transport time is always required for material or energy to move in a process or a plant. It is represented by an e^{-ts} element in a transfer function, where t is a time delay.

In simulation, time delay is usually approximated by a rational function, e.g.,

$$e^{-ts} = \frac{1 - \frac{t}{2}s}{1 + \frac{t}{2}s} \qquad (5.1.4.1)$$

which is known as a 1st order Pade approximation. Higher order Pade approximation up to 3rd order in numerator and fourth order in denominator can be used.

Pade approximation has played an important role in time delay simulation, and in most simulation cases, it gives a good approximation.

Mathematically, the precision of the Pade approximation depends on its order and the amount of time delay. The smaller the time delay is, the better the approximation will be. And the higher the order of Pade approximation is, the more precise mathematically the approximation will be. Usually, more precise mathematical approximation gives more correct simulation results.

In order to analyze the system performance and to design the ideal PID controller parameters, the closed loop response of the system to a unit step input can be obtained, as described below, by solving the closed loop system differential equation with the Pade approximation for time delay.

1. Replace e^{-ts} by the Pade approximation.
2. Develop the closed loop transfer function.
3. Rearrange the closed loop differential equation for easy use.
4. Solve the differential equation with Runge-Kutta method.
5. Plot the system output response to a unit step input.

For the convenience of discussion, consider a typical process model (Model 3). Then the open loop transfer function of PID controller and the process with the 1st order Pade approximation for time delay can be found as

$$G_{OL}(s) = \frac{N_{OL}(s)}{D_{OL}(s)} = \frac{K_O K_C (2 - ts)(t_I s + 1 + t_I t_D s^2)}{[(T_1 s + 1)(T^2 s^2 + 2Txs + 1)(2 + ts)]t_I s} \qquad (5.1.4.2)$$

The closed loop transfer function can be denoted as

$$G_{CL}(s) = \frac{N_{CL}(s)}{D_{CL}(s)} = \frac{N_{OL}(s)}{D_{OL}(s) + N_{OL}(s)} \qquad (5.1.4.3)$$

However, it can be found that the system always becomes unstable if any integral action is added for a system with the Pade approximation of time delay. The smaller the time delay is, more unstable the system is (faster divergence). Even though higher order Pade approximations are used, the simulation results are not improved. The reason for this system unstability can be explained by the root locus of the system. A zero in the right-hand-side s-plane is introduced to the open loop transfer function by the 1st order Pade approximation and the integral action of a PID controller introduces a zero at the origin in the s-plane. Consequently, there is a root locus on the real axis from the origin to the zero in the right-hand-side s-plane as the open loop gain K increases from 0 to infinity.

In order to solve the system unstability problem, the subprogram TDA solve the open loop system differential equation, as described below, with a feedback calculation instead of the Pade approximation of time delay.

1. Develop the open looptransfer function without time delay.
2. Rearrange the no-time-delay open loop differential equation for ease use.
3. Solve the no-time-delay open loop differential equation for one step with Runge-Kutta method.
4. Caculate the system output on the step by a time delay, and then error on the step.
5. Go back to step 3 for the next step if all computation does not finish.
6. Plot the system output response to a unit step input.

The open loop transfer function of PID controller and the process without time delay is

$$G_{OL}(s) = \frac{N_{OL}(s)}{D_{OL}(s)} = \frac{K_O K_C (t_I s + 1 + t_I t_D s^2)}{[(T_1 s + 1)(T^2 s^2 + 2Txs + 1)]t_I s} \qquad (5.1.4.4)$$

It is easy to find that

$$N_{OL}(s) = N_2 s^2 + N_1 s + N_0 \qquad (5.1.4.5)$$

where

$$N_2 = K_O K_C t_D$$
$$N_1 = K_O K_C$$
$$N_0 = K_O K_C / t_I$$

and

$$D_{OL}(s) = D_4 s^4 + D_3 s^3 + D_2 s^2 + s \qquad (5.1.4.6)$$

where

$$D_4 = T_1 T^2$$
$$D_3 = T\,(\,T + 2xT_1\,)$$
$$D_2 = 2xT + T_1$$

By applying the chain rule to the relationship between output C and input U as

$$\frac{C(s)}{E(s)} = \frac{C(s)}{X(s)}\,\frac{X(s)}{E(s)} = \frac{N_{OL}(s)}{D_{OL}(s)} \qquad (5.1.4.7)$$

the relationship can be separated into two part:

$$C(s) = X(s)\,N_{OL}(s) \qquad (5.1.4.8)$$

$$X(s) = E(s) / D_{OL}(s) \qquad (5.1.4.9)$$

It follows that

$$(\,D_4 s^4 + D_3 s^3 + D_2 s^2 + s\,)\,X(s) = E(s) \qquad (5.1.4.10)$$

i.e.,

$$e(t) = D_4 \frac{d^4x}{dt^4} + D_3 \frac{d^3x}{dt^3} + D_2 \frac{d^2x}{dt^2} + \frac{dx}{dt} \tag{5.1.4.11}$$

which may be expressed as

$$\begin{cases} x = & x_1 \\ \frac{dx_1}{dt} = & x_2 \\ \frac{dx_2}{dt} = & x_3 \\ \frac{dx_3}{dt} = & x_4 \\ \frac{dx_4}{dt} = & -\frac{1}{D_4}x_2 - \frac{D_2}{D_4}x_3 - \frac{D_3}{D_4}x_4 + \frac{1}{D_4}e \end{cases} \tag{5.1.4.12}$$

Also,

$$C(s) = (N_2 s^2 + N_1 s + N_0) X(s) \tag{5.1.4.13}$$

or

$$c(t) = N_2 x_3 + N_1 x_2 + N_0 x_1 \tag{5.1.4.14}$$

Choosing step width h in such a way that the time delay is integer times of h, then u(t)=e(t) while t is less than time delay. Assigning the unit step input u(t)=1 (t>0), c(h), c(2h),..., c(t), can be obtained by solving the equation (5.1.4.12) with Runge-Kutta method. The output y(kh) can be also obtained by

$$y(kh) = c(kh-t) \qquad (5.1.4.14)$$

and the error e(kh) is then

$$e(kh) = u(kh) - y(kh) = u(kh) - c(kh-t) \qquad (5.1.4.15)$$

Using Runge-Kutta formulas together with the above feedback relationships, the system response to a step input can be obtained and would be stable for suitable parameters of PID controller to the process.

PCET subprogram TDA was mainly developed by M. Rao and R. Dong, is used to obtain the unit step response of the process, controller and closed loop system. The keystroke sequences required to execute each of these options are summarized in Table 5.1.1.

Example 5.1

Consider a 1st order system with time delay,

$$G(s) = \frac{e^{-s}}{5.2s+1}$$

A PI controller can be designed with TDA by means of trial and error method to make the closed loop system response satisfy the following performance:

Step	User input	Function
1	Type PCET	Start program
2	Select TDA	Time domain analysis
3	Press 1, 2 or 3 for model selection or load a saved file by F3	1. Polynomial model 2. Pole-zero model 3. A typical process model
4	Press 1, 2, 3 or 4	1. Analysis of open loop process 2. Analysis of controller action 3. Analysis of closed loop response 4. Change setpoint/disturbance
5	Input system model and/or controller parameters	Specify system dynamics.
6	Revise input if necessary	Verify input data
7	Select autoscaling, or input figure dimension	Autoscaling computes axis dimension from response data
8		Desired response is displayed on the screen
9	Press a function key to continue	F1-Exit, back to main menu F2-Restart, restart TDA F3-Load, load a saved data file F4-Save, save the data in a file F5-Scale, change the dimension F6-Modify, change the model F7-Tuning, change controller parameters

Table 5.1.1 Functions of TDA in PCET.

overshoot < 0.3

settling time < 10 seconds

One of the most popular semi-empirical controller tuning method is the trial and error tuning method. First, an ultimate gain K_{CU}, at which the system response is a sustained oscillation with a constant amplitude, can be found by making a small proportional gain increment each time with a P controller. Then, take K_C to be $0.5K_{CU}$. To find a critical integral time at which the sustained oscillation comes again with the selected K_C, make a small integral time decrement each time with a PI controller. Then, take integral time t_I equal to 3 times of the critical integral time. Now, small adjustment of parameters can be made to satisfy the given performance. The executions for this example using TDA are illustrated by Figures 5.1.1(a), 5.1.1(b) and 5.1.1(c). The closed loop system response with the designed PI controller is shown in Figure 5.1.1(d).

5.2 Routh Stability Criterion (RSC)

A very important characteristic of a system is its stability. A dynamic system is stable if the system output response is bounded for all bounded inputs, regardless of the initial conditions of the system. Otherwise, the system is unstable.

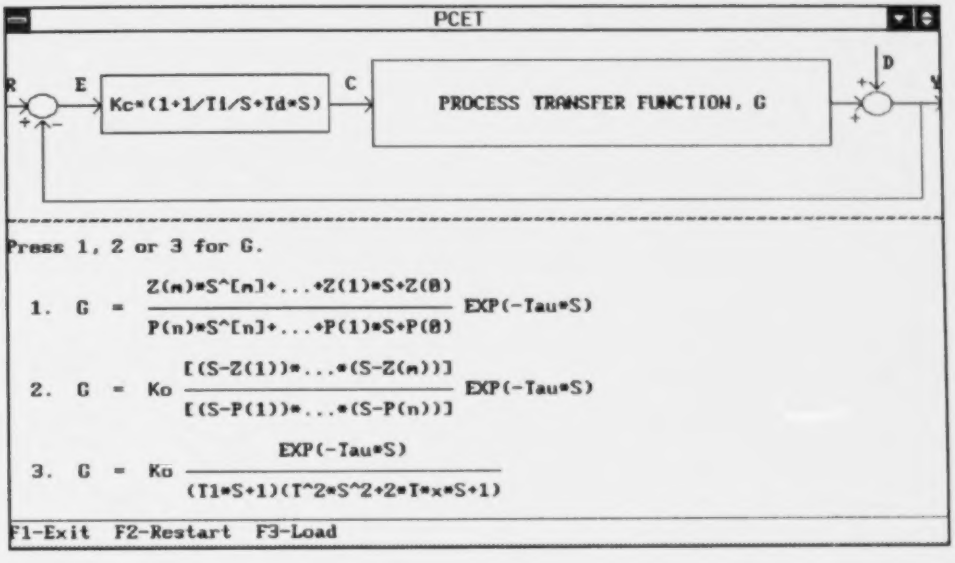


Figure 5.1.1(a) Process model selection of Example 5.1

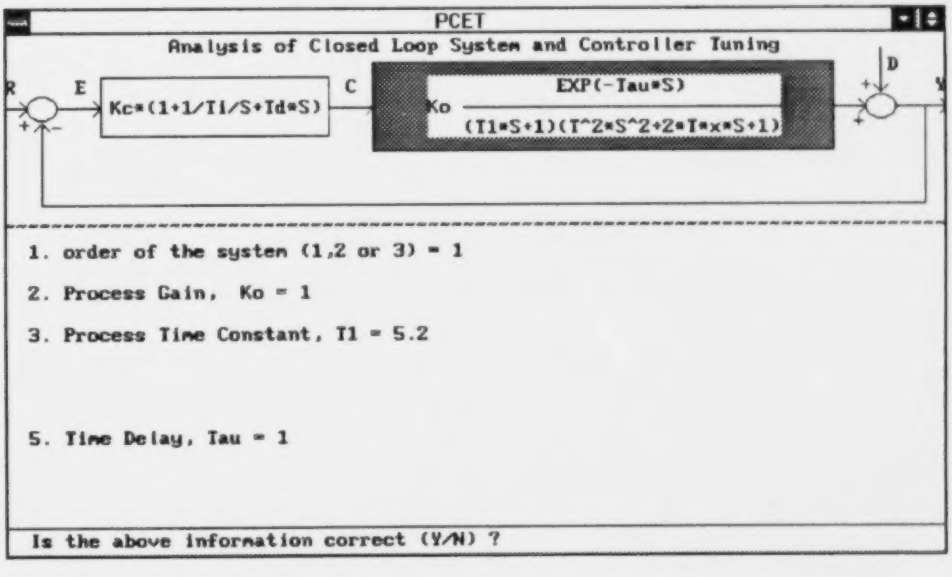

PCET
Analysis of Closed Loop System and Controller Tuning
R
E
Kc*(1+1/TI/S+Id*S)
C
Ko EXP(-Tau*S) / (T1*S+1)(T^2*S^2+2*T*x*S+1)
D
1. order of the systen (1,2 or 3) = 1
2. Process Gain, Ko = 1
3. Process Time Constant, T1 = 5.2
5. Time Delay, Tau = 1
Is the above information correct (Y/N) ?

Figure 5.1.1(b) Process parameters input of Example 5.1

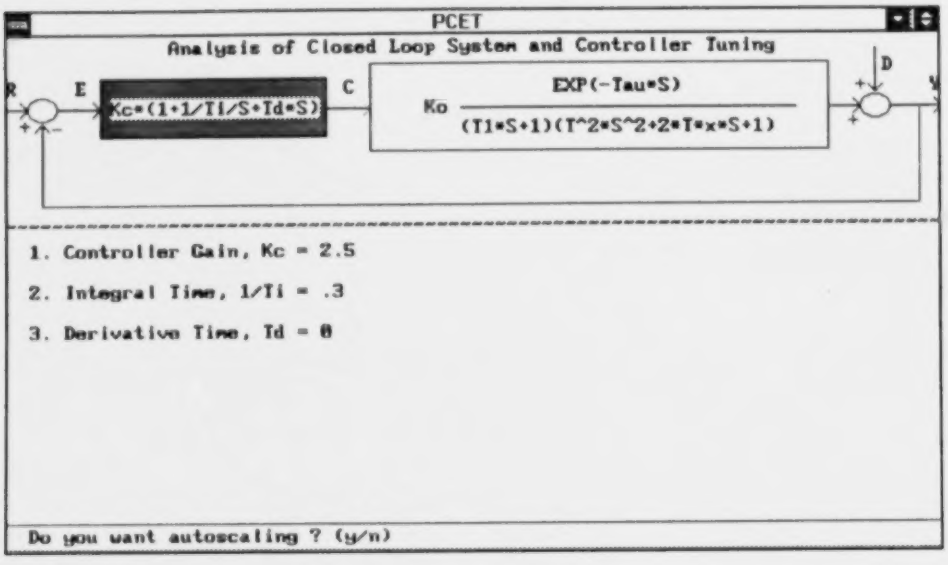


Figure 5.1.1(c) PID controller parameters input of Example 5.1

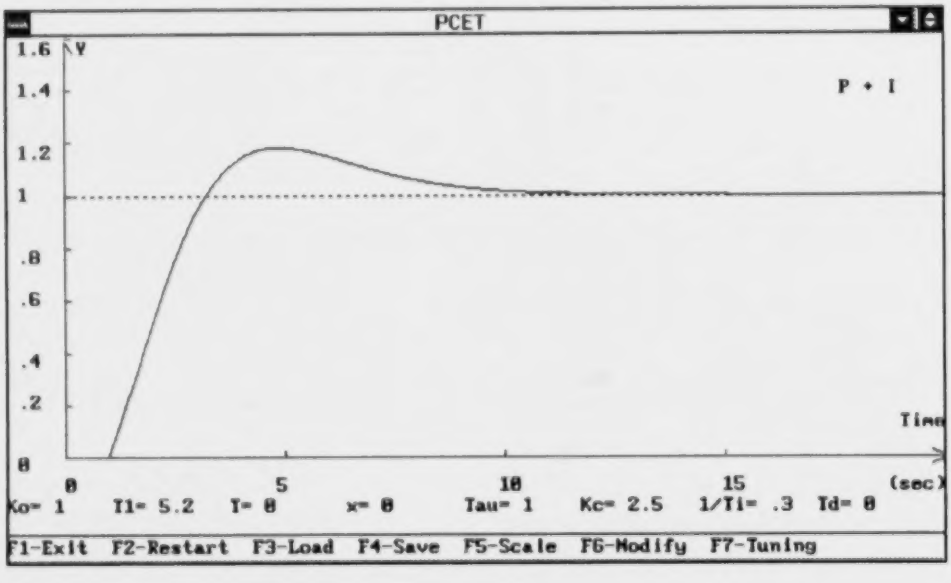


Figure 5.1.1(d) Time domain response of Example 5.1

5.2.1 Characteristic equation and stability

The response of the system is primarily characterized by the denominator of the transfer function. Therefore, the denominator of the system transfer function is referred to as the characteristic polynomial of the system. By setting the characteristic polynomial equal to zero, the characteristic equation can be obtained. The roots of the characteristic equation are referred to as poles of the system. The characteristic equation plays a decisive role in determining system stability.

In general, the system poles are represented by a real part and an imaginary part. The real part affects system stability, while imaginary part determines whether the system response to a step input is oscillatory or not. With respect to stability then, the s-plane can be divided into two regions: stable region (left half s-plane) and unstable region (right half s-plane, including the imaginary axis). The stability of an open loop system is determined by open loop poles, while the stability of a closed loop system is determined by closed loop poles.

5.2.2 Routh Array

If the poles of a system are known, the stability of the system can be easily determined. However, it is often difficult to find the poles by solving a high order algebraic equation. With the Routh criterion, also called the Routh-Hurwitz

criterion, it is not necessary to calculate actual values of the characteristic roots to determine system stability. The criterion provides a way to investigate how many poles of the system are located in the right half s-plane without solving algebraic equations.

For the characteristic equation of a system,

$$a_n s^n + a_{n-1} s^{n-1} + \ldots\ldots + a_1 s + a_0 = 0 \qquad (5.2.2.1)$$

the necessary condition for this system to be stable is that all the coefficients of the system characteristic equation must be positive (have the same sign as a_n). It follows that if any coefficients of the characteristic equation are negative or zero, then at least one root of the characteristic equation lies to the right of, or on, the imaginary axis, and thus the system is unstable. However, it must be noted that it is just a necessary but not sufficient condition. In other words, if all the coefficients of the characteristic equation are positive, stability is not guaranteed.

If all the coefficients of characteristic equation are positive, a Routh array can be constructed as

$$\begin{array}{c|cccc} s^n & a_n & a_{n-2} & a_{n-4} & \cdots \\ s^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & \cdots \\ s^{n-2} & b_1 & b_2 & b_3 & \cdots \\ s^{n-3} & c_1 & c_2 & \cdots & \\ \vdots & \vdots & & & \\ s^0 & h_1 & & & \end{array} \qquad (5.2.2.2)$$

where

$$b_1 = \frac{a_{n-1}a_{n-2} - a_n a_{n-3}}{a_{n-1}} = \frac{-1}{a_{n-1}}\begin{vmatrix} a_n & a_{n-2} \\ a_{n-1} & a_{n-3} \end{vmatrix}$$

$$b_2 = \frac{a_{n-1}a_{n-4} - a_n a_{n-5}}{a_{n-1}} = \frac{-1}{a_{n-1}}\begin{vmatrix} a_n & a_{n-4} \\ a_{n-1} & a_{n-5} \end{vmatrix}$$

$$c_1 = \frac{b_1 a_{n-3} - a_{n-1}b_2}{b_1} = \frac{-1}{b_1}\begin{vmatrix} a_{n-1} & a_{n-3} \\ b_1 & b_2 \end{vmatrix}$$

$$c_2 = \frac{b_1 a_{n-5} - a_{n-1}b_3}{b_1} = \frac{-1}{b_1}\begin{vmatrix} a_{n-1} & a_{n-5} \\ b_1 & b_3 \end{vmatrix}$$

and so on. A Routh array has n + 1 rows. The Routh stability criterion says that the number of roots of a polynomial on the right half s-plane is equal to the number of changes in sign of the first column of the Routh array, i.e., the signs of a_n, a_{n-1}, b_1, c_1, ..., h_1. It follows that a necessary and sufficient condition for a stable system is that all the elements in the first column of the Routh array, constructed by the coefficients of its characteristic equation, are positive.

The Routh criterion is valid only for linear systems. The characteristic equation must be a linear algebraic one, and all the coefficients must be real. The

equation must be a linear algebraic one, and all the coefficients must be real. The Routh criterion is generally qualitatively used to determine only the absolute stability of a system. For two stable systems, the criterion does not indicate which one is more stable, or which one has better dynamic characteristics.

The Routh criterion is also programmed into the software PCET. The sub-program RSC (Routh Stability Criterion) was mainly developed by Y. Ying, and determines Routh array, the roots of the system characteristic equation and then the system stability. The procedure for running this program is shown in Table 5.2.1. If any process transfer function used in other subprograms is loaded by F3, it is automatically find the characteristic equation in this subprogram.

For Step 6 in the procedure of Table 5.2.1, selection of F6, or F7, will display the Routh array, or the roots of the characteristic equation. When a zero appears in the first column of the Routh array, it is replaced by a small positive number epsilon to continue the Routh array. Epsilon is a very small positive number. The value used in RSC is epsilon = 10^{-16}, but this value can be changed by selecting F8.

Example 5.2

If the characteristic equation of a system is

$$s^4 + s^3 + 2s^2 + 2s + 5 = 0$$

its parameters are input into RSC (Figure 5.2.1(a)), the Routh array can be

Step	User input	Function
1	Type PCET	Start program
2	Select RSC	Routh stability criterion
3	Select F3 to input system data or F4 to load a saved file	F1-Main menu F2-Restart F3-Data F4-Load
4	Input system characteristic equation	Specify system dynamics
5	Confirm the correctness of the data	Verify input data
6	Select F6 to get the Routh array or F7 to obtain the characteristic equation roots or F8 to change the epsilon value	F1-Main F2-Restart F3-Data F4-Load F5-Save F6-Routh F7-Root F8-Epsilon

Table 5.2.1 Functions of RSC in PCET.

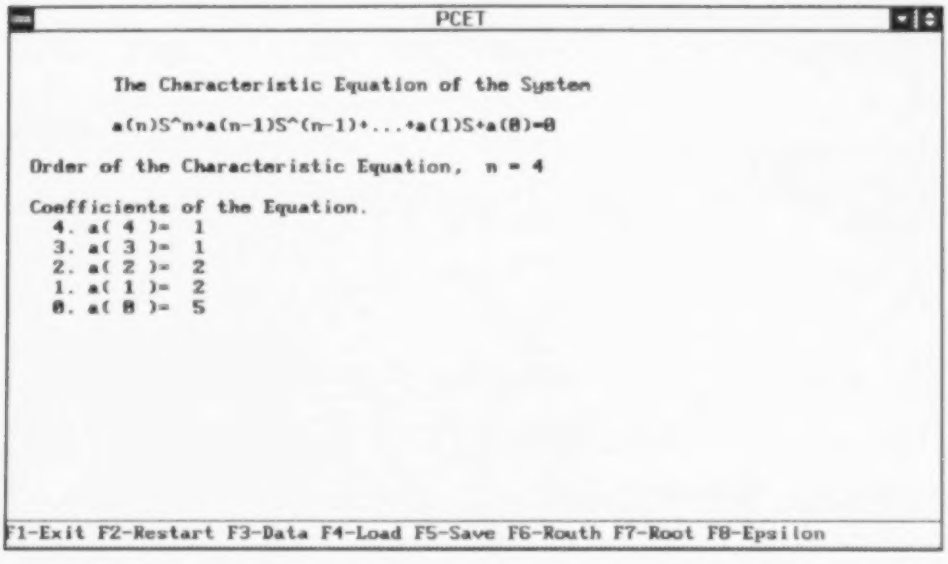

PCET
The Characteristic Equation of the Systen
a(n)S^n+a(n-1)S^(n-1)+...+a(1)S+a(0)=0
Order of the Characteristic Equation, n = 4
Coefficients of the Equation.
4. a(4)= 1
3. a(3)= 1
2. a(2)= 2
1. a(1)= 2
0. a(0)= 5
F1-Exit F2-Restart F3-Data F4-Load F5-Save F6-Routh F7-Root F8-Epsilon

Figure 5.2.1(a) Characteristic equation parameters of Example 5.2

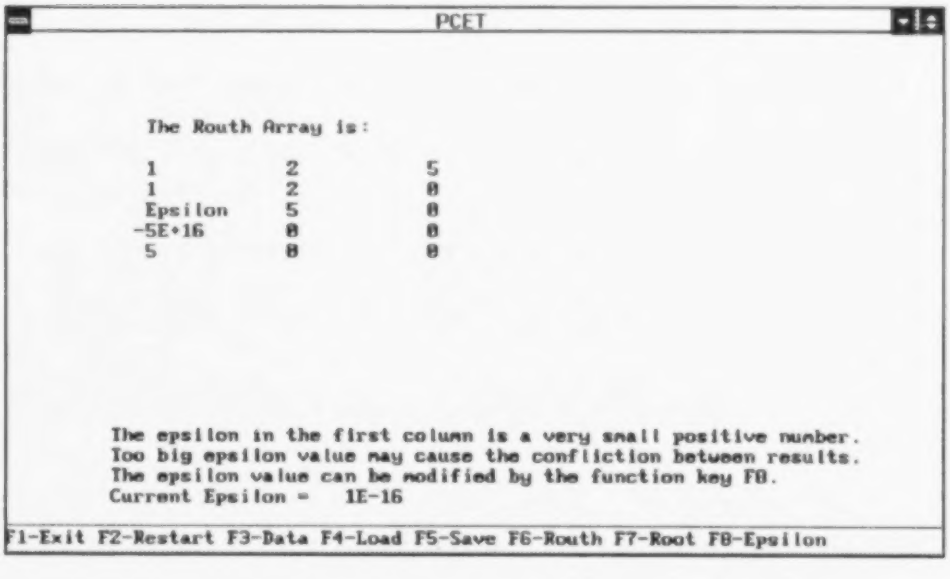

PCET
The Routh Array is:
1 2 5
1 2 0
Epsilon 5 0
-5E+16 0 0
5 0 0
The epsilon in the first column is a very small positive number.
Too big epsilon value may cause the confliction between results.
The epsilon value can be modified by the function key F8.
Current Epsilon = 1E-16
F1-Exit F2-Restart F3-Data F4-Load F5-Save F6-Routh F7-Root F8-Epsilon

Figure 5.2.1(b) Routh array of Example 5.2

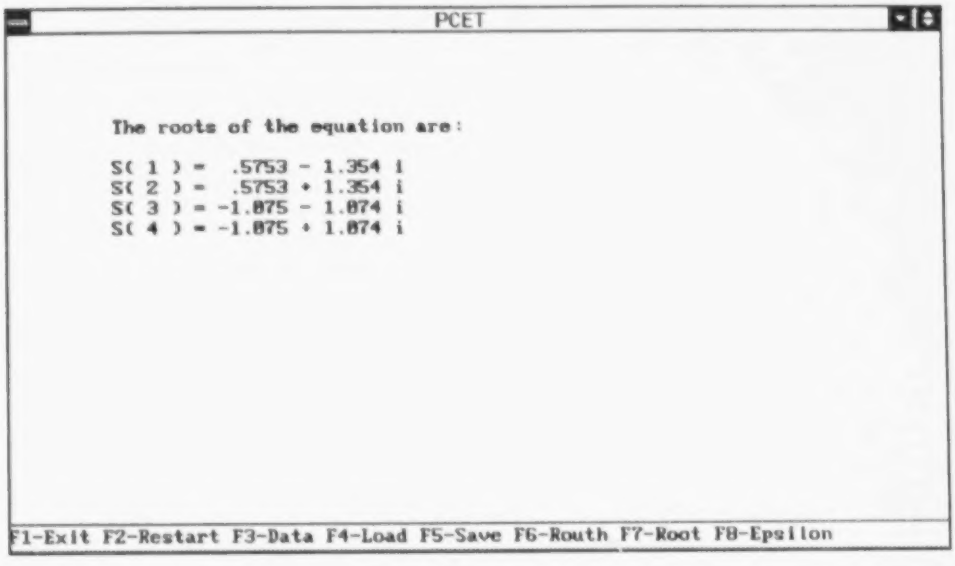

PCET
The roots of the equation are:
S(1) = .5753 - 1.354 i
S(2) = .5753 + 1.354 i
S(3) = -1.875 - 1.874 i
S(4) = -1.875 + 1.874 i
F1-Exit F2-Restart F3-Data F4-Load F5-Save F6-Routh F7-Root F8-Epsilon

Figure 5.2.1(c) Roots of the characteristic equation of Example 5.2

rewritten by the replacement of zero with epsilon in the first column as shown in Figure 5.2.1(b). Since epsilon is a small positive number, 2-5/epsilon is negative, and the number of changes in sign of the first column of the array is 2. Thus, as shown in Figure 5.2.1(c), there are two poles with positive real parts, and the system is unstable.

5.3 Root Locus Technique (RLT)

The stability of a system is of prime consideration for control engineers. The degree of stability, called relative stability, provides information about not only how stable the system is, but also to some extent what dynamics the system exhibits. The relative stability and transient performance of a closed loop control system are directly related to the location of its poles in the s-plane, i.e., the roots of the closed loop characteristic equation. It is useful to determine the locus of the characteristic equation roots in the s-plane as a parameter varies. Root locus is the locus or path of the closed loop roots traced out in the s-plane as a system parameter is changed.

The root locus technique is a graphical method for drawing the root locus. The root locus method provides an approximate sketch of closed loop poles that can be used to obtain qualitative information concerning the stability and performance of the system.

5.3.1 Starting and End Points of the Root Locus

The root locus is usually determined as the open loop gain K varies from zero to infinity. When K = 0, the roots of the characteristic equation are simply the poles of open loop transfer function. When K approaches infinity, the roots of the characteristic equation tend to the zeros of open loop transfer function. This means that the root locus starts at the poles of the open loop transfer function (K = 0) and moves to its zeros as K approaches to infinity. For most common open loop transfer functions, there may be several zeros at infinity in the s-plane. Usually we mark each pole of an open loop transfer function in a root locus as X, and each zero as O.

5.3.2 Points at Imaginary Axis

It is useful to calculate the cross points of the root loci on the imaginary axis. These cross points can indicate when the system will be unstable as the parameter K changes from zero to infinity, and how the system behaves as the system becomes critical.

On the imaginary axis, $s = j\omega$. The cross points of root loci to the imaginary axis must satisfy the closed loop characteristic equation since they are on the loci, and satisfy $s = j\omega$ since they are on the imaginary axis. By substituting $s = j\omega$ into the closed loop characteristic equation, and separating the real and

imaginary parts, two equations can be obtained. The cross points and the corresponding parameter K can be obtained by solving these two equations.

If only the parameter K of the cross points is needed, the Routh criterion is an easy method. The characteristic equation of the system includes the open loop parameter K. The range of values of K that make the system stable can be found by the Routh criterion. The critical value corresponds to the cross points.

In the subprogram RLT, the closed loop roots are directly calculated from the closed loop characteristic equation without usinh the conventional root locus technique that was developed for hand drawing.

5.3.3 System Analysis and Design with Root Locus

For a single loop feedback control system, the selection of an open loop gain is essentially a P controller design. Thus, the root locus may be used for the analysis and design of a system with a P controller. In fact, the root locus can also be used in the analysis and design of the system with other controllers, such as a PD controller, PI controller and so on.

The root locus of a system can be plotted with the subprogram RLT in the software package PCET. The subprogram RLT was mainly developed by H. Zhou. The open loop transfer function may be written in the polynomial form or

zero-pole form, both of which are acceptable for the software. The procedure for plotting the root locus on the screen with PCET is shown in Table 5.3.1.

Example 5.3

The open loop transfer function of a system is given by

$$G_{OL}(s) = \frac{K_O}{(s+0.2)(s+0.4)}$$

With a PI controller, the equivalent transfer function may be found to be

$$\hat{G}(s) = \frac{K/t_I}{s[(s+0.2)(s+0.4)+K]}$$

where $K = K_O K_C$.

When the controller has only a proportional action, the root locus of the open loop transfer function is shown in Figure 5.3.1(b). With a decay ratio of 0.25, K may be obtained as 1.86. Then, the equivalent transfer function becomes

$$\hat{G}(s) = \frac{1.86/t_I}{s[(s+0.2)(s+0.4)+1.86]}$$

The root locus with respect to the integral time is shown in Fig. 5.3.1(d), where it should be noted that the root locus traces to infinity as the integral time decreases to zero rather than going to infinity.

Step	User input	Function
1	Type PCET	Start program
2	Select RLT	Root locus technique
3	Press 1,2 or 3 for model selection	1. Polynomial model 2. Pole-zero model 3. A typical process model
4	Input maximum gain Kmax	The root locus will be drawn as the open loop gain K varies from 0 to the maximum gain Kmax.
5	Input system model parameters	The order and parameters of the system model are determined.
6	Revise input if necessary	Verify input data.
7		Root locus is displayed on the screen.
8	Select F1 or F2 to change Kmax, F3 or F4 to get gain or frequency at imaginary axis, F5 or F6 to obtain closed loop roots or open loop zeros, or F7 to finish analysis and be ready for another system.	F1-Decrease maximum K F2-Increase maximum K F3-Gain at Imag. Axis F4-Frequency at Imag. Axis F5-Closed Loop Roots F6-Open Loop Zeros F7-End

Table 5.3.1 Functions of RLT in PCET.

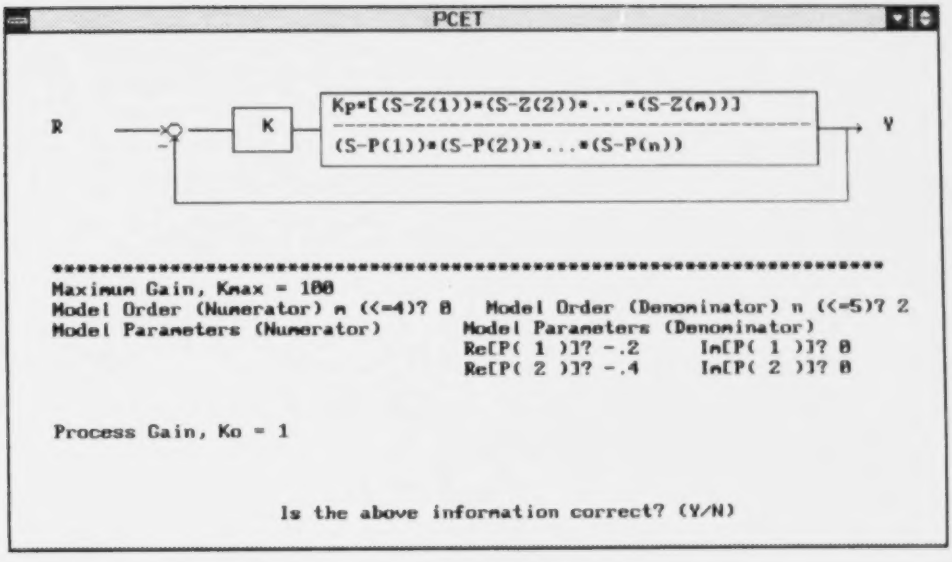


Figure 5.3.1(a) Open loop transfer function without controller of Example 5.3

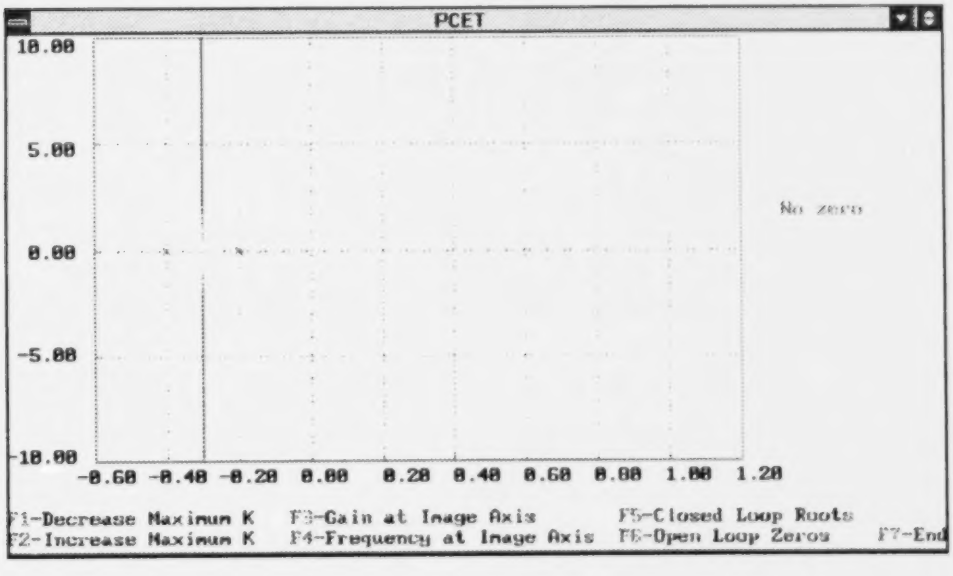


Figure 5.3.1(b) Root locus with respect to proportional gain of Example 5.3

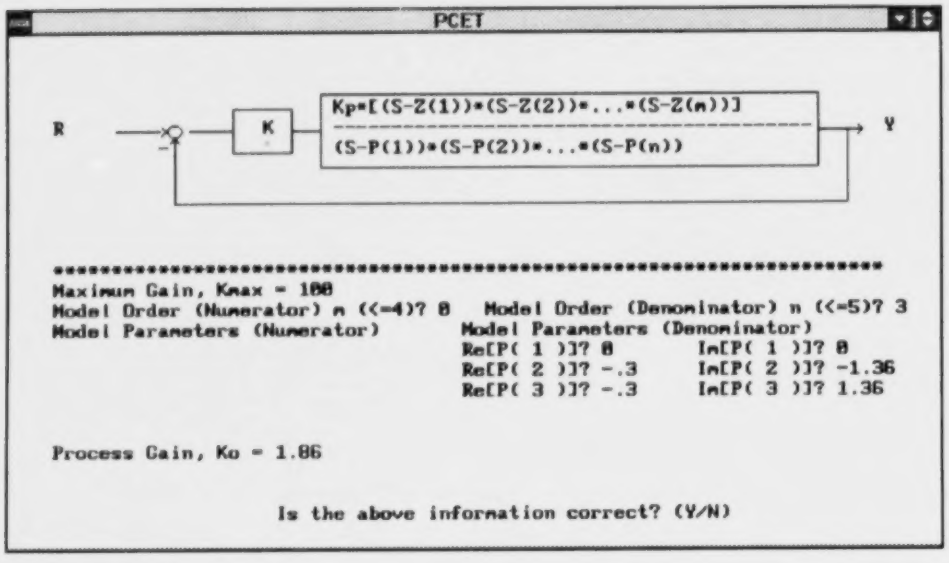


Figure 5.3.1(c) Equivalent transfer function with PI controller of Example 5.3

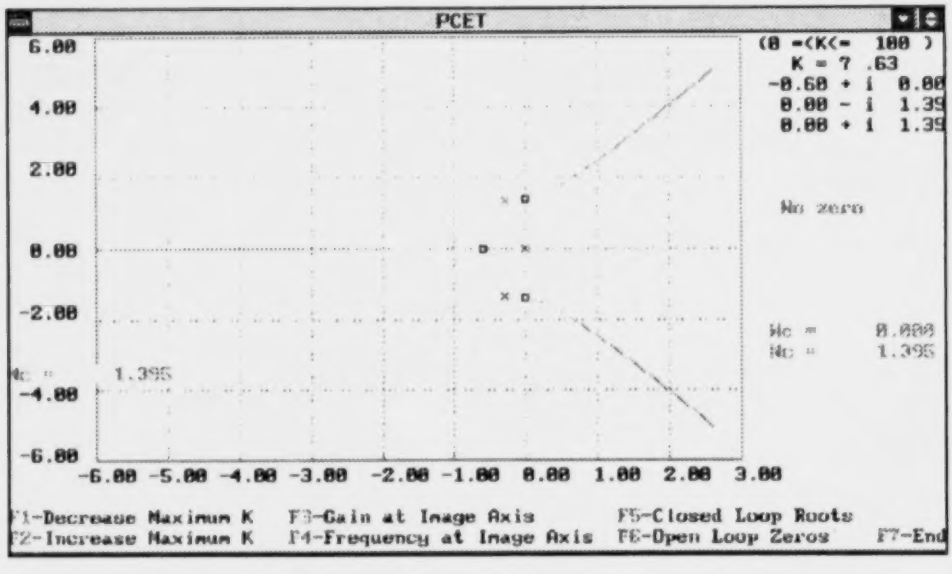


Figure 5.3.1(d) Root locus with respect to t_1 of Example 5.3

5.4 Frequency Domain Analysis (FDA)

In reality, a system rarely incurs a single disturbance but rather a series of disturbances. To simulate those disturbances, sinusoidal signals with specified frequencies can be applied. In other words, they can be analyze in the frequency domain. The Nyquist plot and Bode diagram are two basic frequency responses of a system, and both are means of analyzing the stability and dynamics of a system in the frequency domain.

5.4.1 Sinusoidal Signals

When a linear system is subjected to a sinusoidal input, its ultimate response is also a sustained wave. This important fact forms the basis of frequency response analysis. The sinusoidal input is generated by the expression

$$f(t) = \begin{cases} A\sin wt & t \geq 0 \\ 0 & t < 0 \end{cases} \qquad (5.4.1.1)$$

The parameters A and ω fix the amplitude and angular frequency of the oscillation. The period T of a sine wave is related to its angular frequency by $T = 2\pi/\omega$. The Laplace transform of a sinusoidal function is

$$F(s) = \frac{A\omega}{s^2 + \omega^2} \qquad (5.4.2.2)$$

5.4.2 Frequency Response

The frequency response of a system is defined as the ultimate response of the system to a sinusoidal input signal. Frequency response is easy to determine. For a linear system, the ultimate state of any signal in the system is sinusoidal with the identical frequency as the sinusoidal input. For a sinusoidal signal, there are three elements: frequency, amplitude and phase angle. As a matter of fact, if the response is determined by experiment, only the amplitude and phase shift of the output need to be measured for every designated frequency. If the transfer function G(s) of the system is known, the frequency response can be determined simply by calculating the amplitude ratio and phase shift of $G(j\omega)$.

Both frequency response and transient response reflect the performance of a system, but except for some special cases the correlation between them is not straightforward. However, a system that has a satisfactory frequency response will normally have a satisfactory transient response, or vice versa.

5.4.3 Nyquist Plot

A Nyquist plot is the polar plot of frequency response $G(j\omega)$ as the angular frequency ω varies from 0 to infinity. This plot is used to represent the system characteristics in the frequency domain. Nyquist plots use the imaginary part of $G(j\omega)$, denoted $\mathrm{Im}[G(j\omega)]$, as the ordinate, and the real part of $G(j\omega)$, denoted

Re[G(jω)], as the abscissa. The shape and location of a Nyquist plot reflect the characteristics of the system.

Nyquist plots can be drawn on the screen or printed out with the sub-program FDA (Frequency Domain Analysis) of the software package PCET. Three different input transfer function models can be selected: the polynomial model, the factored polynomial model and the zero-pole model. After typing the order and parameters of the model as well as the maximum and minimum frequency of the plot, the Nyquist plot is obtained by assuming that a P controller is used with Kc=1.

5.4.4 Bode Diagram

The Bode diagram or Bode plot is a plot of amplitude ratio (AR) and phase shift (ϕ) of G(jω) with respect to the angular frequency ω.

The phase shift is usually plotted against the logarithm of frequency on semilog coordinates, i.e., the abscissa is dimensioned by log(ω). With such a dimension, the distance between 0.1 and 1 is equal to that between 1 and 10, or between 10 and 100, and so on. The distance between 1 and 10, or an equivalence, is called a decade.

The amplitude ratio is also plotted against the logarithm of angular fre-

quency in semilog coordinates. For convenience, the amplitude ratio is usually converted to log modulus (logarithmic modulus) defined by the equation

$$L = \text{log modulus} = 20 \log|G(j\omega)| \tag{5.4.4.1}$$

The units of log modulus are decibels (dB), a term originally used in communication engineering to indicate the ratio of two values of power. However, some control engineers plot the amplitude ratio in log-log coordinates, with dimensions in real values instead of log modulus. In fact, it is just the same plot with a different scale.

The stability of a system is of prime consideration for control engineers. Also important is the relative stability, which gives the degree of stability and to some extent, system dynamics. The root locus or frequency response can be used to investigate the relative stability of a system.

The gain and phase margins of a system can be easily evaluated from its Bode diagram. The unit circle in a polar plot corresponds to the 0 dB line in the log modulus plot of a Bode diagram, and the negative real axis corresponds to the -180 degree line in a phase shift plot. Since the gain margin is defined as

$$20\log GM = 0 \text{ dB} - 20\log|G(j\omega_c)| \text{ dB} \tag{5.4.4.2}$$

the gain margin on a Bode diagram is the distance in dB between the 0 dB line

and the point on the log modulus curve at critical frequency ω_c. If the 0 dB line is above the point on the log modulus at critical frequency, the gain margin is positive, and the closed loop system is stable. However, for a stable system, the log modulus at critical frequency is negative, while the gain margin is positive as mentioned above. The phase margin on a Bode diagram is the distance in degrees between the -180 degree line and the point on the phase shift curve at gain crossover frequency. If the -180 degree line is above the point on the phase shift curve, the phase margin is negative, and the closed loop system is unstable. Otherwise, the phase margin is positive, and the system is stable. However, it should be noted that the critical frequency may not exist, e.g., for a first order system.

For a single loop feedback control system, the open loop transfer function is G(s), whereas the closed loop transfer function is G(s)/[1 + G(s)]. In the time domain, the system stability is investigated by checking the closed loop characteristic equation to find whether or not all the poles are located on the left half s-plane. In the frequency domain, however, the closed loop system stability can be determined on the basis of the open loop frequency response G(jω) rather than the closed loop one. The gain margin and phase margin are checked on the open loop frequency response to determine the stability of the closed loop system.

Bode diagrams can also be drawn on the screen or printed out with the subprogram FDA in the software package PCET. Since the first several steps for drawing a Bode diagram on screen are the same as those for drawing a Nyquist

diagram, the procedure for plotting both Bode diagrams and Nyquist plots can be found in Table 5.4.1. Since the computer screen is limited, the log modulus and the phase shift are overlapped on the screen. The left side dimension is for log modulus, and the right side for phase shift. The stability margin can be displayed on the screen after the Bode diagram is drawn.

Gain margin and phase margin are two important measures of relative stability to be obtained from the frequency response of a system. They can also be taken as performance criteria of a system for analysis and design, since they affect not only the relative stability but also the dynamics of the closed loop system. Several Bode diagrams should be drawn when designing a controller. Furthermore, the proposed controller may not satisfy the design requirement, and may need to be adjusted to achieve the best performance. The subprogram FDA of PCET can easily draw the frequency response and simulate the controller tuning. First, a Bode diagram of the open loop transfer function can be obtained on the screen. Then, through trial and error, a satisfactory PID controller can be found by tuning controller parameters with F9.

Example 5.4

Consider a single loop feedback control system with the process transfer function of

$$G(s) = \frac{e^{-0.5s}}{(s+1)(2s+1)}$$

Step	User input	Function
1	Type PCET	Start program
2	Select FDA	Frequency domain analysis
3	Press 1,2 or 3 for model selection or load a saved file by F3	1. Polynomial model 2. Factored polynomial model 3. Zero-pole model.
4	Input system model parameters	Define the system and and determine the frequency range.
5	Select F6 for the Nyquist plot or F7 for the Bode diagram	F1-Exit F2-Restart F3-Load F4-Save F5-Modify F6-Nyquist F7-Bode
6	If the Nyquist plot is on screen, F8 changes the display demension. If the Bode diagram is on, F8 displays gain and phase margin. Select F9 to tune PID controller parameters.	F1-Exit F2-Restart F3-Load F4-Save F5-Modify F6-Nyquist F7-Bode F8-Scale or Margin F9-PID

Table 5.4.1 Functions of FDA in PCET.

To design P, PI and PID controllers respectively with the same 10dB gain margin, using the subprogram FDA in PCET, the open loop Nyquist plot and Bode diagram of the system, as shown in Figure 5.4.1(b) and 5.4.1(c) respectively, can be obtained with a proportional controller $K_C = 1$. User input screen is shown in Figure 5.4.1(a). The critical frequency ω_c can be found as about 1.8. Tuning the P controller to get gain margin 10dB, Figure 5.4.1(d), Bode diagram of the system with phase margin 69.23 can be obtained. The P controller parameter is K_C=2.136. With the empirical formula $t_I=(3\text{-}6)/\omega_c$, $1/t_I=0.5$ can be chosen. With trial and error method and using FDA to get 10 dB for a PI controller, Figure 5.4.1(e), the Bode diagram of the system with the PI controller can be obtained, where a phase margin 42.8 can be found. The parameters of the PI controller are $K_C = 1.36$ and $1/t_I=0.5$. With the empirical formula $t_I=(3\text{-}6)/\omega_c$ and $t_D=(0.25\text{-}0.35)t_I$, $1/t_I=0.5$ and $t_D=0.5$ can be chosen. With trial and error method and using FDA to get 10dB for a PID controller, Figure 5.4.1(f), the Bode diagram of the system with the PID controller can be obtained, where a phase margin 41.61 can be found. The parameters of the PID controller are K_C=3.4, $1/t_I=0.5$ and $t_D=0.5$

In this example, the gain margin are the same 10dB. The system with PID controller has the smallest phase margin, while the system with a P controller has the largest one. The response of the system with three different controllers are shown in Figure 5.4.1(g), which is obtained by using the subprogram TDA. It can be realized that the system with PID controller has a better dynamics while the system with a P controller has steady state error.

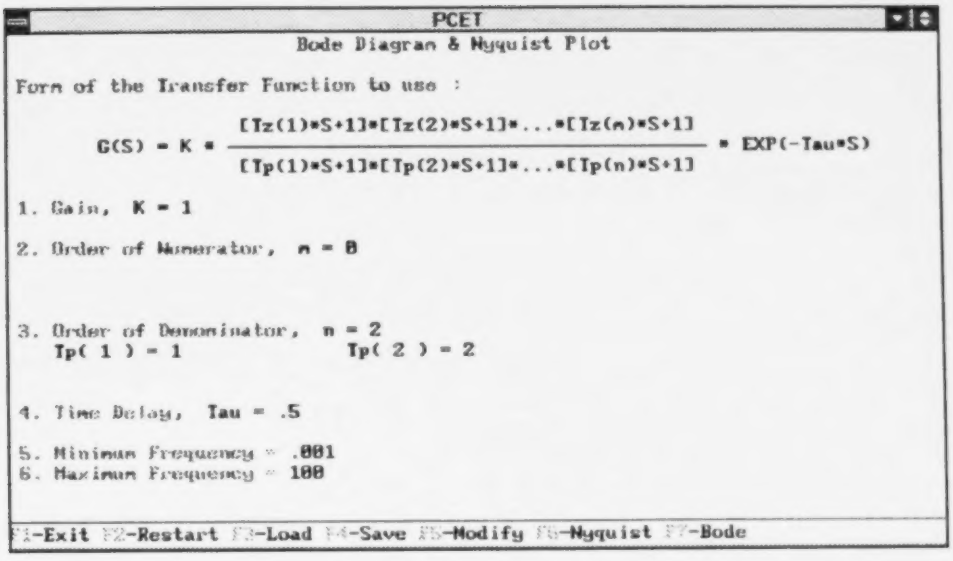

PCET
Bode Diagram & Nyquist Plot
Form of the Transfer Function to use :
G(S) = K * [Tz(1)*S+1]*[Tz(2)*S+1]*...*[Tz(n)*S+1] / [Tp(1)*S+1]*[Tp(2)*S+1]*...*[Tp(n)*S+1] * EXP(-Tau*S)
1. Gain, K = 1
2. Order of Numerator, n = 0
3. Order of Denominator, n = 2
Tp(1) = 1
Tp(2) = 2
4. Time Delay, Tau = .5
5. Minimum Frequency = .001
6. Maximum Frequency = 100
F1-Exit F2-Restart F3-Load F4-Save F5-Modify F6-Nyquist F7-Bode

Figure 5.4.1(a) Process model and its parameters of Example 5.4

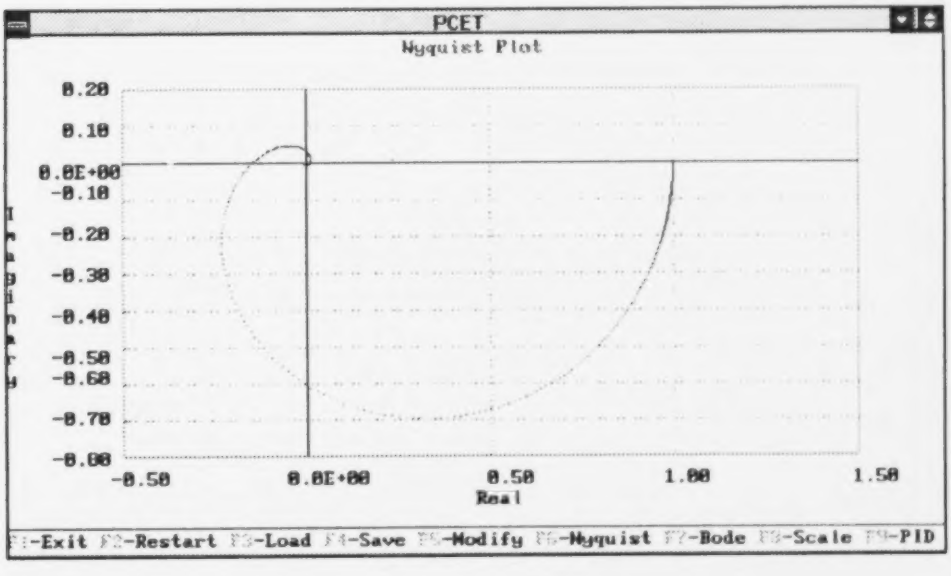


Figure 5.4.1(b) Nyquist plot without controller of Example 5.4

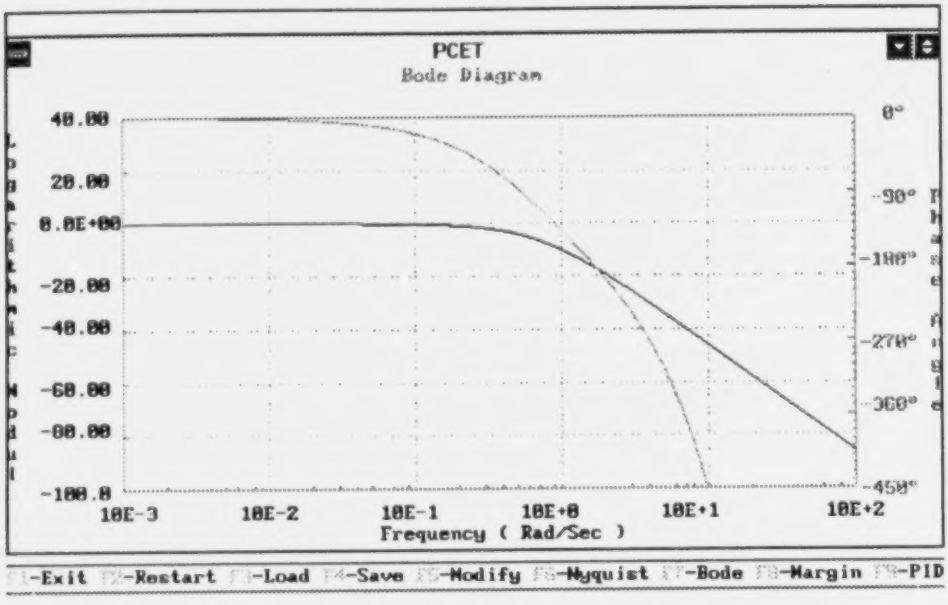


Figure 5.4.1(c) Bode diagram without controller of Example 5.4

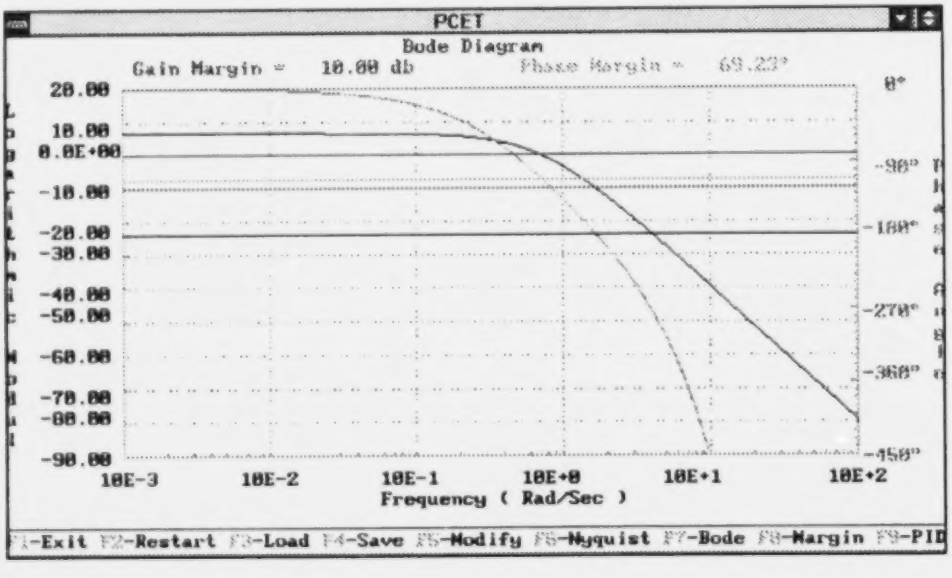


Figure 5.4.1(d) Gain and phase margin with P controller of Example 5.4

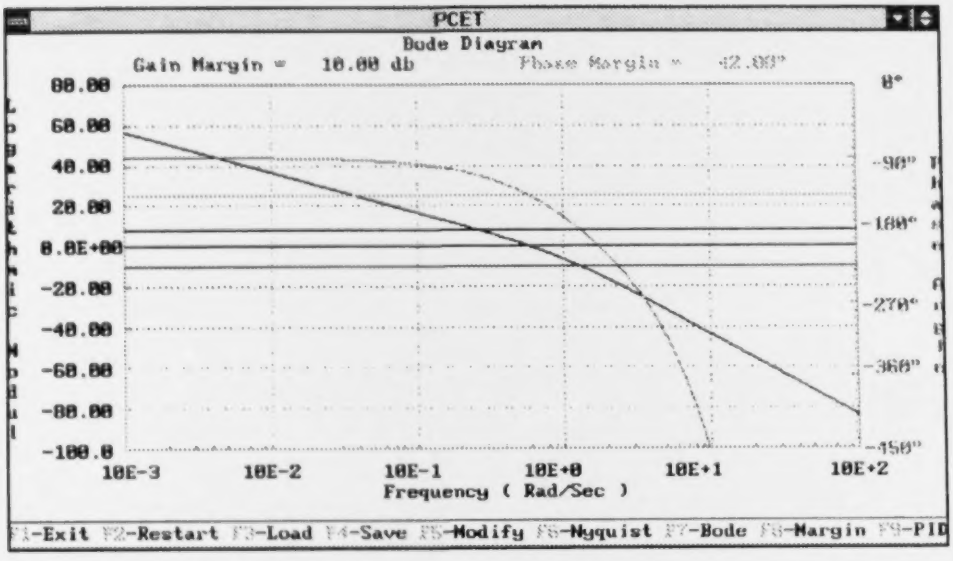


Figure 5.4.1(e) Gain and phase margin with PI controller of Example 5.4

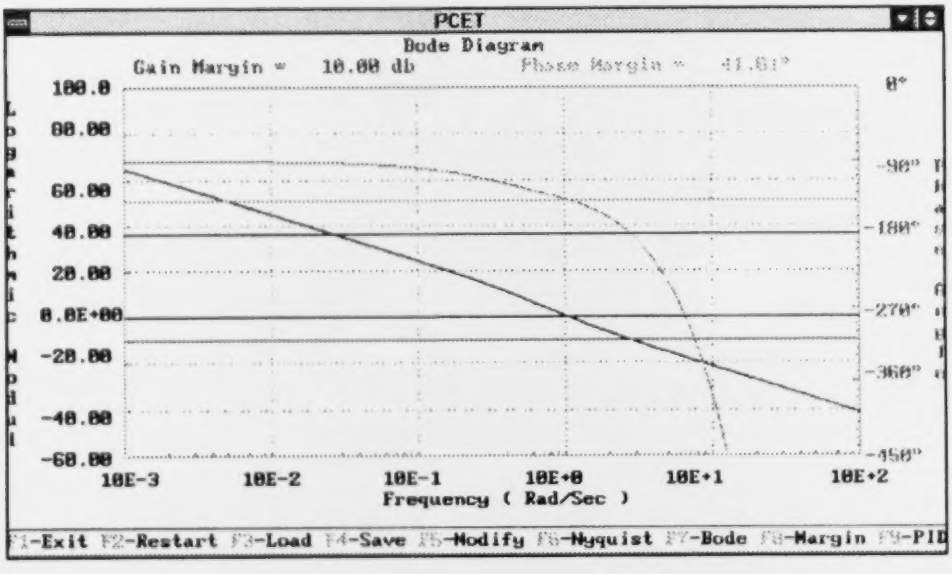


Figure 5.4.1(f) Gain and phase margin with PID controller of Example 5.4

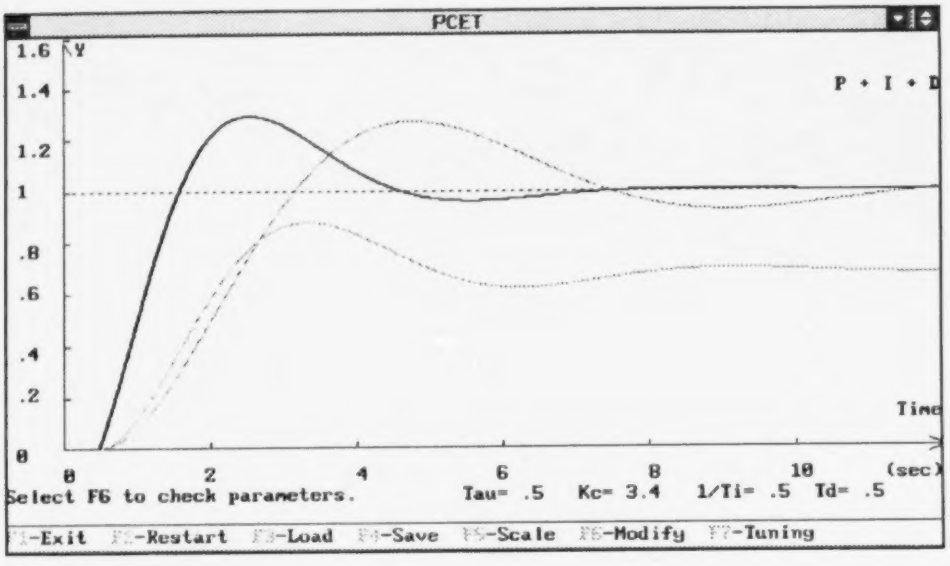


Figure 5.4.1(g) Time domain responses wirh P, PI and PID of Example 5.4

5.5 Discrete-time System Analysis (DSA)

With the dramatic development of digital computers in the past few decades, computer control systems have been used in many applications. In digital computers, digital signals, rather than continuous ones, are used.

5.5.1 Zero-order Hold

Discrete time systems, or sampled data systems, are dynamic systems in which one or more signals are discontinuous or discrete, and can change only at discrete instants of time. Usually, conversion between continuous signals and discrete signals is necessary in a discrete time system.

The conversion from a discrete signal to a continuous signal is made by a holding device. The simplest holding device is a zero order hold, which holds the discrete signal constant at $r(kt_s)$ until a new sample occurs at $t = (k+1)t_s$. The resulting signal is a staircase approximation of the input signal r(t).

By the definition of a zero order hold, its transfer function is the Laplace transform of impulse response and is formulated as

$$H_0(s) = \frac{1 - e^{-t_s s}}{s} \qquad (5.5.1.1)$$

The precision of conversion is limited, and related to the sampling period. The output c(t) of the zero order hold approaches the system input r(t) as the sampling period t_s approaches zero.

5.5.2 Discrete-time System Responses

Before analyzing the responses of discrete systems, the discrete system models must be determined. Discrete systems can be described by difference equations in the time domain or by transfer functions in the z-domain.

For discrete systems, the values of signals at sampling instants are only important. While differential equations are used to describe continuous system dynamics, difference equations are used for discrete systems. The difference equation has a general form

$$y(t) + a_1 y(t-1) + \ldots + a_m y(t-m) = b_0 r(t) + b_1 r(t-1) + \ldots + b_k r(t-k) \qquad (5.5.2.1)$$

where k is less than m. Taking z-transform on both sides and rearranging it yields

$$\frac{Y(z)}{R(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_k z^{-k}}{1 + a_1 z^{-1} + \cdots + a_m z^{-m}} \qquad (k<m) \qquad (5.2.2)$$

This is the transfer function of a discrete system, which represents the relationship between the input and output of the system, and is referred to as a pulse transfer

function. The transfer function in the z-domain is analogous to that used in the s-domain. This input-output model can be used to determine the response of a discrete system to a specified input.

The responses of a sampled system to a specified input can be calculated from the difference equation of the system, which can be obtained from the output in the z-domain. Usually, z-transforms are more complicated than their corresponding Laplace transforms. Fortunately, there are some relatively simple techniques for obtaining inverse z-transforms. The long division method is an easier way to find an inverse z-transform than partial fraction expansion. However, the result is usually in series form, which is not as useful as an analytical expression.

5.5.3 Digital PID Controller

Both continuous and discrete controllers can be used in discrete control systems. In the case of a continuous controller used in a discrete system, the discrete transfer function of the controller is meaningless. The continuous transfer function of the controller should be combined with the process transfer function. Only the discrete transfer function of their combination is meaningful. Therefore, in most discrete systems, discrete rather than continuous controllers are used. In a computer controlled system, the digital controller is an algorithm implemented in the computer.

Through numerical approximations of the integral and derivative parts in

the transfer function of a continuous PID controller, the digital PID controller transfer function can be found as

$$G_C(z) = K_C\left[1 + \frac{t_S}{t_I}\left(\frac{1}{1-z^{-1}}\right) + \frac{t_D}{t_S}\left(1-z^{-1}\right)\right] \qquad (5.5.3.1)$$

where t_S is the sampling period. This is also referred to as the position form of the PID control algorithm since the actual controller output is calculated.

Incorporated in the software package PCET is the subprogram DSA, which will enable the user to examine the dynamics of a unit feedback discrete time system. In this subprogram, only typical process model is considered. The system configuration is shown in Figure 5.5.1, where a digital PID controller and a zero order hold are used.

The sampling period can be assigned by the user. The selection of the sampling period is an important but complicated issue. A number of empirical rules for this selection have been reported. For example, Shannon's sampling theorem can be used to determine the sampling period. However, it is more of an art than a science. The response of a system may react unfavorably to an unsuitably selected sampling period. When an integral action or a derivative action is included in the system, the selection of a sampling period becomes more complicated. Adjusting the parameters of a PID controller may require adjusting the sampling period. One must be aware that if the sampling period is too small, or the dimension scaled is too large, simulation error may happen, since too many

calculation steps produce a large accumulated error.

Similar to TDA for time domain analysis, the subprogram DSA used in the z-domain can be applied for open loop process analysis, controller analysis, closed loop system analysis and controller tuning. The discrete time response of the process or closed loop system to a unit step input or disturbance can be drawn on the screen or with a printer. The keystroke sequences required to obtain the responses are illustrated in Table 5.5.1.

Example 5.5

Consider a discrete system with the third order transfer function

$$G(s) = \frac{4}{(3s+1)(4s^2+4s+1)}$$

A digital PID controller is to be designed so that the closed loop system satisfies the time domain performance criteria:

Overshoot < 0.25

Settling Time < 50 seconds

The sampling period, t_S, is assumed to be 1 second. With the trial and error method by using the subprogram DSA in PCET, a digital PID controller can be found to satisfy the requirement. The parameters of the controller are $K_C = 0.01$, $1/t_I = 2.5$ and $t_D = 7$ (Figure 5.5.2(a), (b) and (c)).

Step	User input	Function
1	Type PCET	Start program
2	Select DSA	Discrete-time system analysis
3	Press 1, 2, 3 or 4	1. Analysis of open loop process 2. Analysis of controller action 3. Analysis of closed loop response 4. Change setpoint/disturbance
4	Input system model, sampling time and/or controller parameters	Specify system dynamics.
5	Revise input if necessary	Verify input data
6	Select autoscaling, or input figure dimension	Autoscaling computes axis dimension from response data
7		Desired response is displayed on the screen
8	Press a function key to continue	F1-Exit, back to main menu F2-Restart, restart DSA F3-Load, load a saved data file F4-Save, save the data in a file F5-Scale, change the dimension F6-Modify, change the model F7-Tuning, change controller parameters

Table 5.5.1 Functions of DSA in PCET.

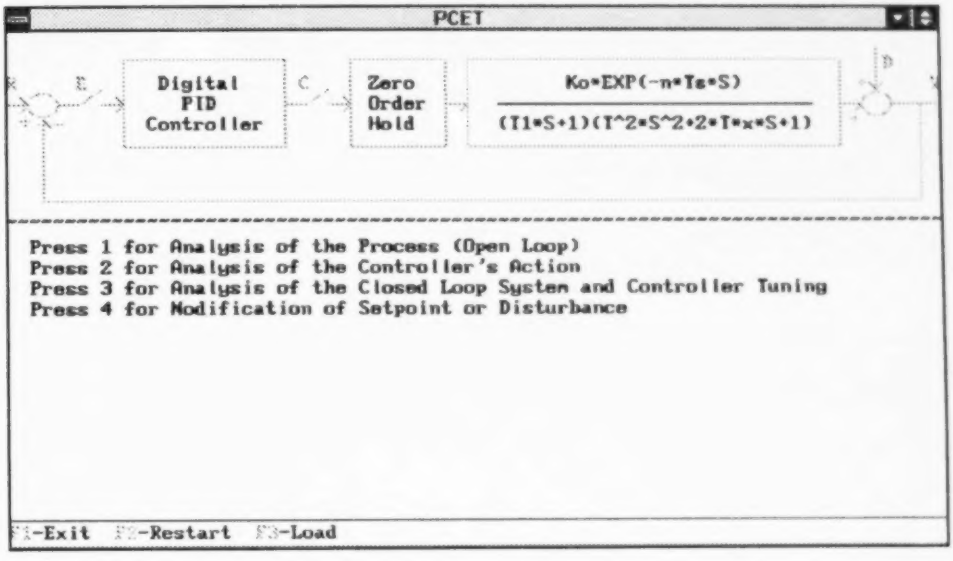


Figure 5.5.1 Discrete-time system configuration in DSA

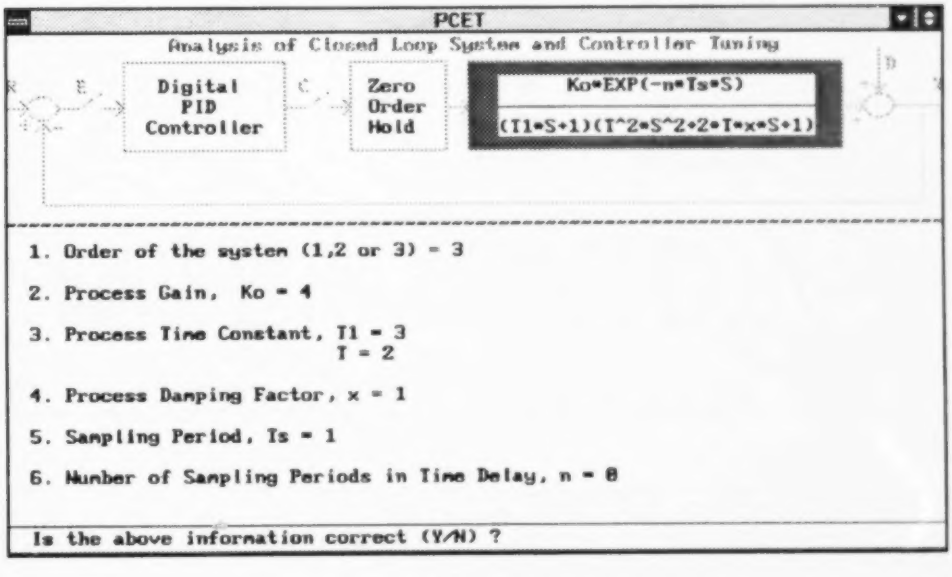


Figure 5.5.2(a) Discrete-time system parameters of Example 5.5

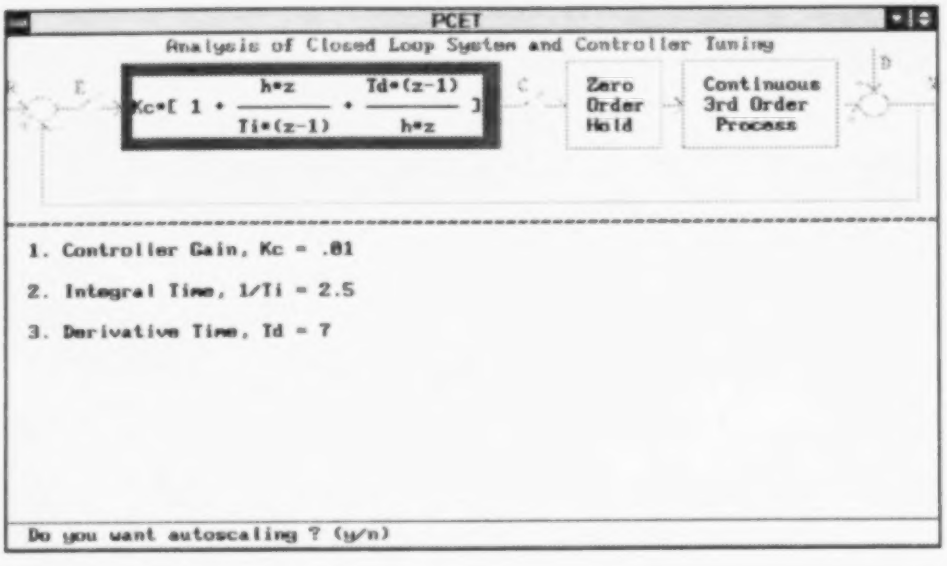

PCET
Analysis of Closed Loop System and Controller Tuning
Kc*[1 + h*z / Ti*(z-1) + Td*(z-1) / h*z]
Zero Order Hold
Continuous 3rd Order Process
1. Controller Gain, Kc = .01
2. Integral Time, 1/Ti = 2.5
3. Derivative Time, Td = 7
Do you want autoscaling ? (y/n)

Figure 5.5.2(b) Digital PID controller parameters of Example 5.5

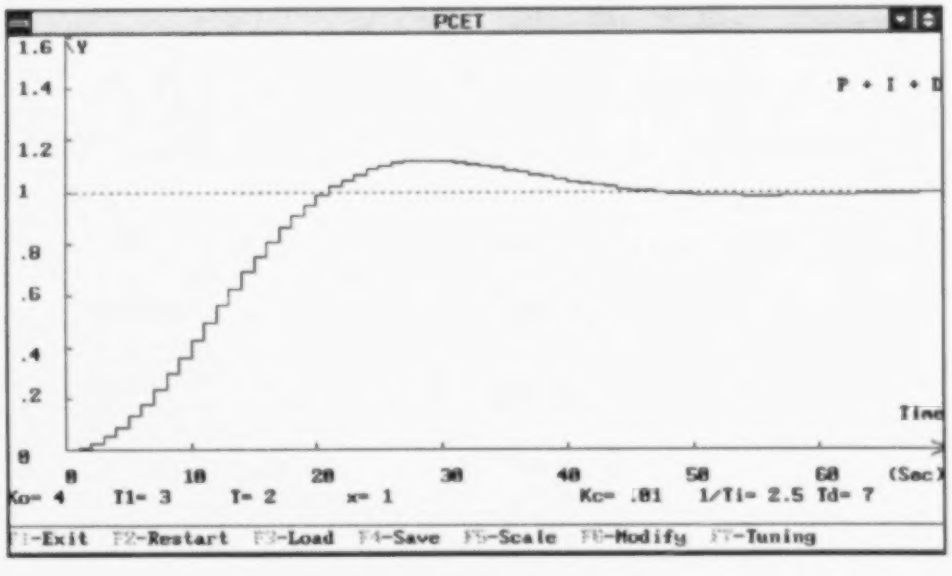


Figure 5.5.2(c) Discrete-time response of Example 5.5

The resulting response is shown in Figure 5.5.2(c). According to the empirical formula for the choice of sampling period, $0.1t_D<t_s<0.5t_D$ and $t_s>0.001t_r$, the choice of the sampling time of 1 second is reasonable.

5.6 Linear State Space Analysis (LSA)

Conventional control theory is generally applicable only to linear time-invariant systems having a single input and a single output. The state space control theory is a useful tool for studying systems with multiple inputs and multiple outputs and/or time-variant systems. For classification, throughout this Section 5, capital bold letters and lowercase bold letters represent matrices and vectors respectively.

5.6.1 State Space

In conventional control theory, only the inputs, outputs and actuating errors are considered important. Besides inputs, outputs and actuating errors, there are other variables in a control system which can also reflect the system behavior or states, and which may be described in a system model.

The state of a dynamic system is the smallest set of system variables that, together with the inputs, can completely determine the system behavior. With this

definition, the state of the system at time t is determined by the system state at time t_0 and the input for $t \geq t_0$. The variables that determine the system state are referred to as state variables. All the state variables of a dynamic system, which can completely describe the system behavior, can construct a vector that is called a state vector.

If a system can be completely described by n state variables, the n-dimensional space whose n coordinate axes represent the n state variables is called a state space. The state of the system at any time can be represented by a point (a state vector) in the state space. The system state will trace a trajectory in the state space while time t varies.

5.6.2 State space representation

A linear time-invariant system may be described by a linear constant differential equation. Defining n new state variables and rearranging the sets of equations in vector form, the state space representation of the system takes the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{5.6.2.1}$$

where **x**, **y** and **u** are state, output and input vectors, respectively. **A**, **B**, **C** and **D** are compatible constant matrices.

5.6.3 Controllability and Observability

There are two basic problems to consider. The first one is the coupling between the input and the state: Can any state be controlled by the input? This is a controllability problem. Another is the relationship between the state and the output: Can all the information about the state be observed from the output? This is an observability problem.

5.6.3.1 Controllability Criterion

A system is said to be controllable at time t_0 if it is possible to find an unconstrained control vector to transfer any initial state to the origin in a finite time interval. Stated mathematically, the system is controllable at t_0 if for any $x(t_0)$, there exists $u[t_0, t_1]$ that gives $x(t_1) = 0$ $(t_1 > t_0)$. If this is true for all initial time t_0 and all initial states $x(t_0)$, the system is completely controllable.

With the above definition and the Cayley-Hamilton theorem, the necessary and sufficient condition for the system to have complete controllability is that the controllability matrix,

$$\mathbf{P} = [\mathbf{B}|\mathbf{AB}|\mathbf{A}^2\mathbf{B}|\ \ldots\ |\mathbf{A}^{n-1}\mathbf{B}] \qquad (5.6.3.1.1)$$

is of full rank.

5.6.3.2 Observability Criterion

A system is said to be observable at time t_0 if it is possible to determine the state $x(t_0)$ from the output function over a finite time interval. In mathematical terms, the system is observable at t_0 if any $x(t_0)$ can be estimated by the observation of $y[t_0, t_1]$ $(t_1>t_0)$. If this is true for all time t_0 and all states $x(t_0)$, the system is completely observable.

Since observability is the coupling between the state variables and the output of a system, for observability discussion, the system input can be assumed to be zero. For the general system, the necessary and sufficient condition of a linear system for complete observability is that the observability matrix,

$$\mathbf{Q} = [\mathbf{C}|\mathbf{A}^T\mathbf{C}|(\mathbf{A}^T)^2\mathbf{C}| \ldots |(\mathbf{A}^T)^{n-1}\mathbf{C}]^T \qquad (5.6.3.2.1)$$

is of full rank.

5.6.4 Diagonalization of matrices

If an n x n matrix **A** has n distinct eigenvalues, n distinct eigenvectors can be found. A modal matrix of matrix **A** consists of the eigenvectors of **A**. If the eigenvectors of an n x n matrix **A** are $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_n$, then, the modal matrix **M** is

$$\mathbf{M} = [\ \mathbf{m}_1,\ \mathbf{m}_2,\ \ldots,\ \mathbf{m}_n\ ]$$

A matrix **A** can be diagonalized by the modal matrix and its inverse:

$$\mathbf{J} = \mathbf{M}^{-1}\ \mathbf{A}\ \mathbf{M}$$

where the diagonal matrix **J** has the eigenvalues of **A** on the diagonal.

If $\mathbf{B}^{-1}\mathbf{AB} = \mathbf{C}$, the relationship between **A** and **C** is called a similarity transformation, and **A** and **C** are similar matrices. Two similar matrices have the same eigenvalues. If a square matrix **A** has multiple eigenvalues, a Jordan canonical form matrix instead of a diagonal matrix can be obtained. The modal matrix **M** consists of the eigenvectors and the general eigenvectors that can be obtained from the formulas

$$(\lambda_i \mathbf{I} - \mathbf{A})\mathbf{x}_{i1} = \mathbf{0}$$

and

$$\mathbf{x}_{ij} = (\lambda_i \mathbf{I} - \mathbf{A})\mathbf{x}_{i(j-1)}$$

where $\mathbf{x}_{i1}$ is the eigenvector associated with the eigenvalue λ_i.

If the eigenvalues of system matrix **A** are distinct, the system can be diagonalized by similarity transformation through the relations

$$\dot{\mathbf{z}} = \mathbf{M}^{-1}\mathbf{A}\mathbf{M}\mathbf{z} + \mathbf{M}^{-1}\mathbf{B}\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{M}$$

where $\mathbf{x} = \mathbf{M}\mathbf{z}$, $\mathbf{M}$ is the modal matrix of $\mathbf{A}$, and $\mathbf{M}^{-1}\mathbf{A}\mathbf{M}$ is a diagonal matrix. When the system matrix becomes diagonal, all the states are decoupled. If one row in $\mathbf{M}^{-1}\mathbf{B}$ is zero, the corresponding state variable cannot be changed by the input. In other words, for a linear system with distinct eigenvalues, the system is completely controllable if and only if no row of $\mathbf{M}^{-1}\mathbf{B}$ has all zero elements. If any column in $\mathbf{C}\mathbf{M}$ is zero, then it is impossible to get any information about the corresponding state variable from the output. This means that for a linear system with distinct eigenvalues, the system is completely observable if and only if no column of $\mathbf{C}\mathbf{M}$ has all zero elements. If the eigenvalues of matrix $\mathbf{A}$ are not distinct, diagonalization may be impossible. By similarity transform, a Jordan canonical form can be obtained.

The PCET software package will enable the user to examine the controllability and observability of a linear system with state space representation by using the subprogram LSA. The subprogram LSA was mainly developed by M. Sardaga. Detail functions in LSA are described in Table 5.6.1.

Example 5.6

Consider a linear system

Step	User input	Function
1	Type PCET	Start program
2	Select LSA	Linear state space analysis
3	Define system dimension and options	Database file name Title of case study Number of state variables Number of input variables Number of output variables Field width for data input Initial guess for eigenvalue Tolerance for convergence
4	Enter data for matrices **A**, **B**, **C** and **D** or load a case study from the database.	Input data of matrices **A**, **B**, **C** and **D**.
5	Display results on screen.	Controllability matrix Observability matrix Rank of controllability matrix Rank of observability matrix Conclusion Eigenvalues of **A** Modal matrix Inverse of modal matrix Transformed matrices **A**, **B**, **C** and **D**
6	Print results	All items in step 5 are printed on printer.
7	Return to main menu.	

Table 5.6.1 Functions of LSA in PCET.

$$\dot{x} = \begin{pmatrix} -6 & -11 & -6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u$$

$$y = [-1 \quad 0 \quad 1]x$$

Controllability and observability matrices are shown in Figure 5.6.1(c) and Figure 5.6.1(d). The modal matrix and its inverse can been found by LSA as shown in Figure 5.6.1(f). The transferred system is also shown in Figure 5.6.1(g). Thus, the system is completely controllable, but not completely observable.

5.7 Industrial Application Case (IAC)

A paper machine headbox is chosen as a typical industrial case study in order to combine theory with practice, and to link abstract mathematical models with the real industrial process. Paper machine headbox control system is widely used in pulp and paper mills to control basis weight and moisture of the products. The main problems encountered with headbox control are the ability to maintain the total pressure and level in the headbox, and to maintain the rush/drag ratio when wire speed changes. The benefits from implementation of a good headbox control system would include (Figure 5.7.1(a)):

1. improvement of wet end stability
2. better and more uniform formation

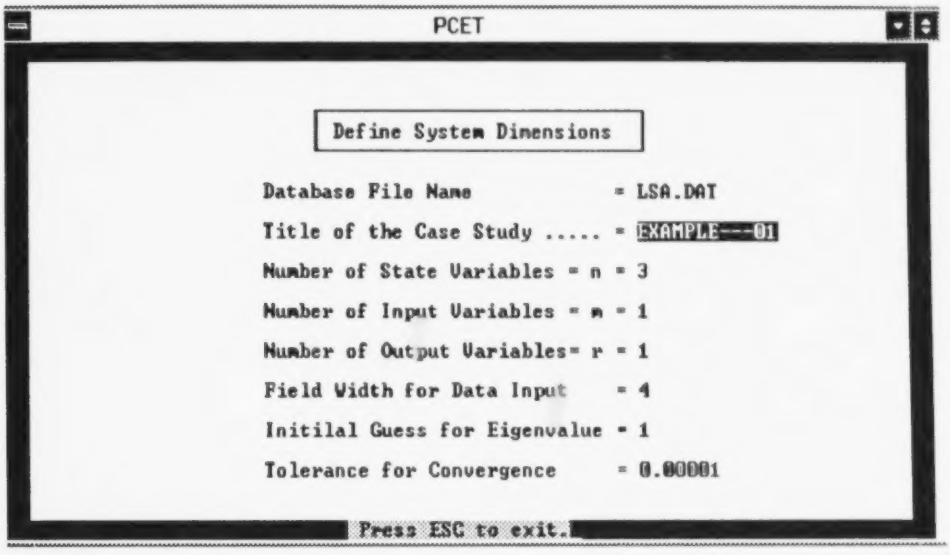

PCET
Define System Dimensions
Database File Name = LSA.DAT
Title of the Case Study ..... = EXAMPLE---01
Number of State Variables = n = 3
Number of Input Variables = m = 1
Number of Output Variables= r = 1
Field Width for Data Input = 4
Initilal Guess for Eigenvalue = 1
Tolerance for Convergence = 0.00001
Press ESC to exit.

Figure 5.6.1(a) System dimension of Example 5.6

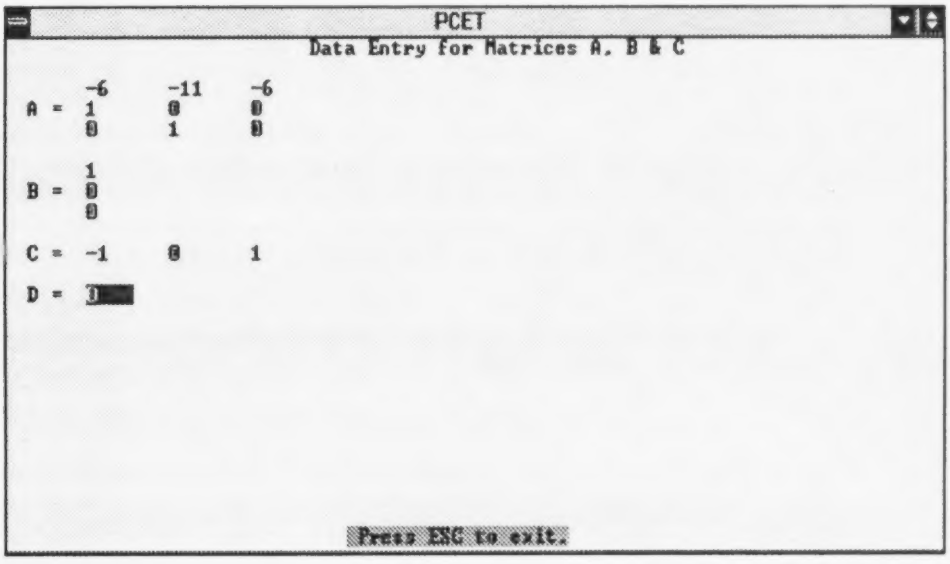


Figure 5.6.1(b) System parameters of Example 5.6

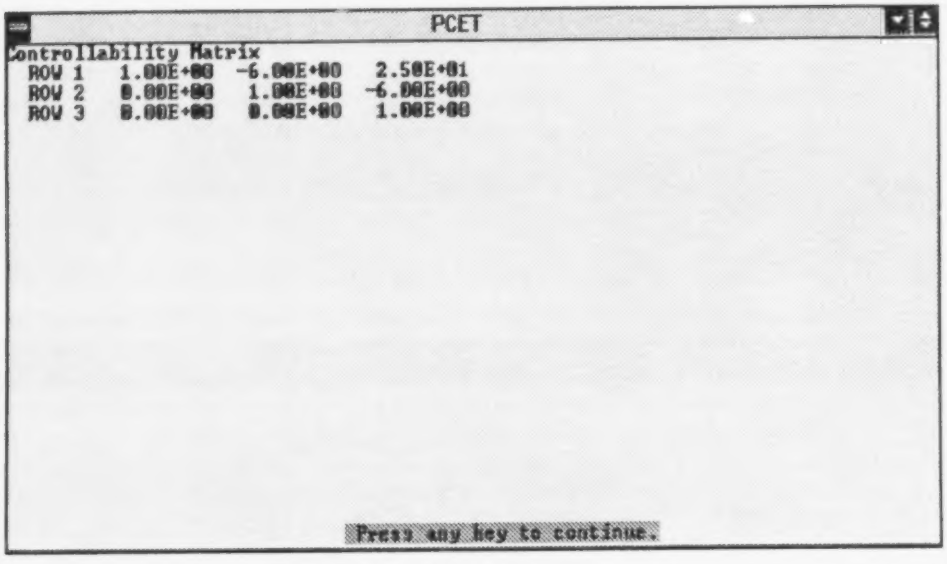


Figure 5.6.1(c) Controllability matrix of Example 5.6

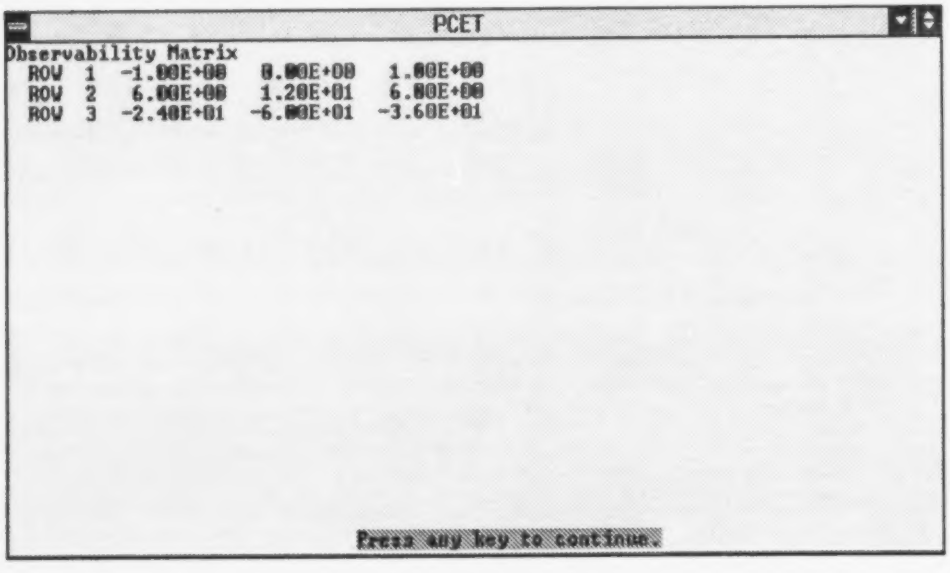


Figure 5.6.1(d) Observability matrix of Example 5.6

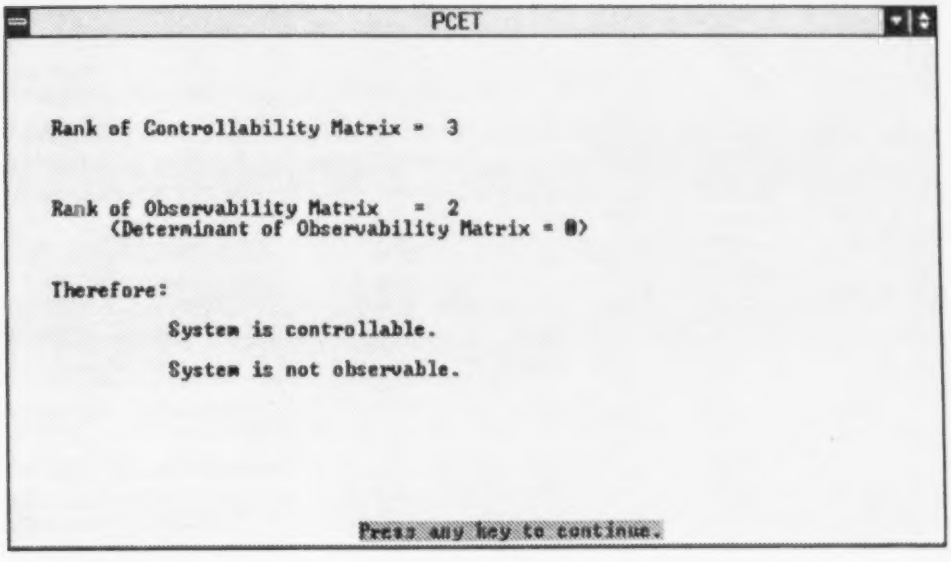

PCET
Rank of Controllability Matrix = 3
Rank of Observability Matrix = 2
(Determinant of Observability Matrix = 0)
Therefore:
System is controllable.
System is not observable.
Press any key to continue.

Figure 5.6.1(e) Controllability anf obserbability analysis of Example 5.6

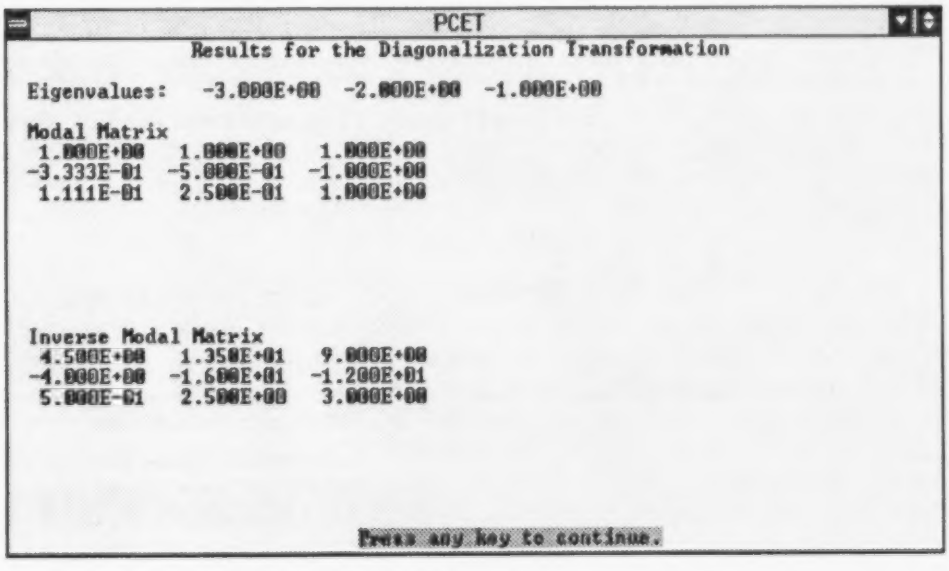


Figure 5.6.1(f) Modal matrix of Example 5.6

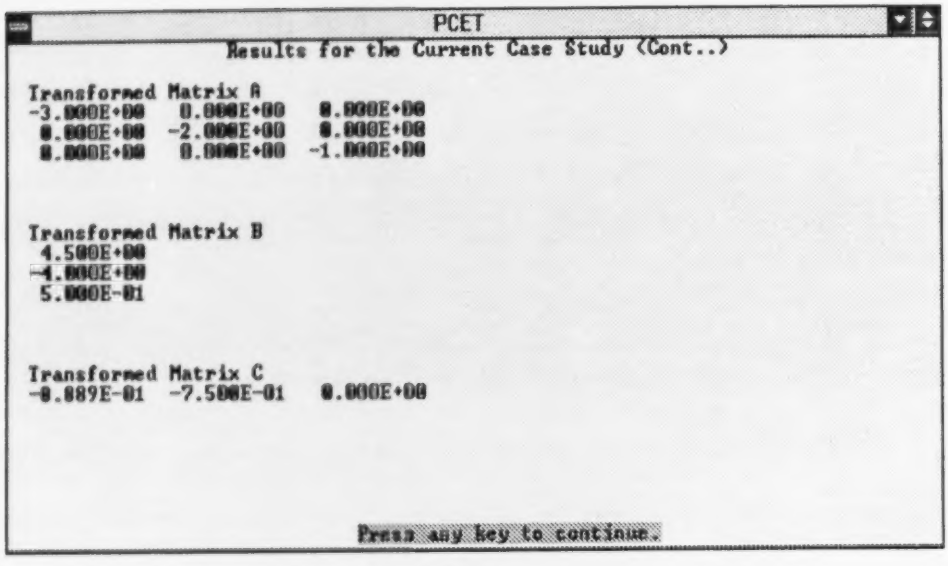


Figure 5.6.1(g) Diagonalized system of Example 5.6

3. automatic wet end speed change capability
4. automatic wet end start-up capability
5. reduction of variations in final sheet quality variables such as tensile and tear ratio in machine direction (MD) and cross direction (CD), stretch, etc.

Different headbox control schemes should be chosen according to the different headbox types and stock delivery systems.

With a model of the paper machine headbox control system, several typical industrial process control configurations, such as single-loop control and cascade control, are also incorporated in PCET. Under each control configuration, several subprograms are linked. They currently includes Time Domain Analysis (TDA) and Routh Stability Criterion (RSC).

5.7.1 Cascade Control

In order to eliminate the effect of disturbances and improve control system's dynamic performance, a common control configuration is taken with two controllers. When the output of one controller is used to manipulate the setpoint of another controller, the two controllers are said to be cascaded, and the system is referred to as a cascade control system (Figure 5.7.1(b)).

The block diagram of a general cascade system is shown in Figure 5.7.1(c), where the number 1 stands for the primary loop, 2 for the secondary loop, p for

process, v for valve, c for controller, m for measurement.

The transfer function of the secondary closed loop is

$$G_2(s) = \frac{G_{C_2}(s)G_v(s)G_{P_2}(s)}{1 + G_{C_2}(s)G_v(s)G_{P_2}(s)G_{m_2}(s)}$$

Note that if $G_{m2}(s) = 0$ and $G_{C2}(s) = 1$, the transfer functions of $G_2(s)$ is reduced to $G_v(s)G_{P2}(s)$ for the corresponding simple feedback system, respectively.

Subprogram IAC (Industrial Application Case) in PCET deals with industrial processes. A paper machine headbox process is typically incorporated. With this package, single loop control and cascade control systems can be analyzed. The procedure for analyzing the cascade control systems is described in Table 5.7.1. The subprogram TDA is incorporated with IAC. The secondary loop response is obtained first. The secondary (closed) loop is automatically dealt with as an element block in the primary loop, when the primary loop response is drawn. When the same procedure used for closed loop analysis in TDA described before, the system responses for both secondary loop and primary loop can be obtained. The input can be selected as a setpoint or disturbance by pressing "4" in Step 4 of Table 5.7.1. The default values are R(setpoint) = 1 and D(disturbance) = 0. When observing the responses to the disturbance in the secondary loop, the setpoint should be set to zero and disturbance can be set to 1 for secondary loop analysis, and both the setpoint and disturbance in the primary loop should be set to zero

Step	User input	Function
1	Type PCET	Start program
2	Select IAC	Industrial application case
3		Headbox control system introduction
4	Select F2 for single loop control or F3 for cascade control	F1-Exit F2-Single F3-Cascade F4-Next
5		Schematic diagram of cascade control system for paper machine headbox
6	Select F4 to display the block diagram or F2 to link TDA for secondary loop response.	F1-Exit F2-Response F3-Help F4-Block
7	Described in Table 5.1.1 to use TDA	Tune the seconday controller, and then exit from TDA. Information for primary loop control
8	Described in Table 5.1.1 to use TDA	Tune the primary controller and get the overall system response.
9	If necessary, repeat step 6 through step 8.	

Table 5.7.1 Functions of IAC in PCET.

for primary loop analysis.

Example 5.7

A paper machine headbox control system is an industrial application case incorporated in PCET. A pressurized headbox control system usually contains total head control and level control loops. The total head is controlled by adjusting thin stock flow rate while the level is controlled by adjusting air pad pressure. The fluctuations in stock flow rate may be caused by many factors including underdamped pulsation from pumps, rotating screens, etc. Therefore, cascade control may be used to eliminate or to minimize the disturbances before they reach the headbox.

A schematic diagram for cascade control configuration, as shown in Figure 5.7.1(b), is provided in the subprogram IAC in PCET. Its block diagram can also be obtained as shown in Figure 5.7.1(c). The tuning of the two controllers in the cascade control system proceeds in two steps.

In the first step, to determine the settings for the secondary controller, the tuning methods for single loop control system can be used. In pulp and paper industries, proportional controller is generally applied for secondary loop. The proportional gain for flow rate system is usually at range of 1-3. Assuming the secondary loop model has the first order transfer function $G_{p_2}(s)$ as

$$G_{P_2}(s) = \frac{1}{8s+1}$$

with the proportional controller of $G_{c2}(s) = 3$, the second loop response to a unit step change in setpoint can be obtained as shown in Figure 5.7.1(e). Since the secondary controller is a proportional controller, the offset exists in the response.

In the second step, to determine the settings for the primary controller, the closed secondary loop is considered as a block in the primary loop, and then the primary controller can be tuned. In practice, PI or PID controller is generally used to eliminate the offset in the control system. Assuming the transfer function of the primary process $G_{P1}(s)$ as

$$G_{P_1}(s) = \frac{2}{2s+1}$$

with a PI controller $G_{c1}(s)$ for the primary loop, the primary loop response to a unit step change in setpoint can be obtained as shown in Figure 5.7.1(g), where the offset no longer exists. The parameters of the primary controller is $K_C = 2$ and $1/t_I = 0.25$. If the response is not satisfactory, the controller parameters can be easily tuned to get the satisfactory response.

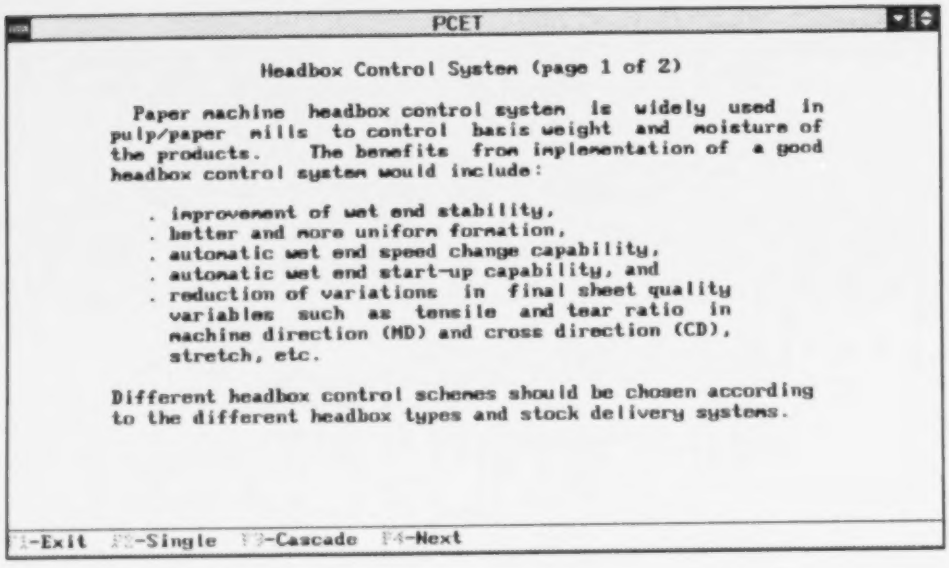

PCET
Headbox Control System (page 1 of 2)
Paper machine headbox control system is widely used in pulp/paper mills to control basis weight and moisture of the products. The benefits from implementation of a good headbox control system would include:
. improvement of wet end stability,
. better and more uniform formation,
. automatic wet end speed change capability,
. automatic wet end start-up capability, and
. reduction of variations in final sheet quality variables such as tensile and tear ratio in machine direction (MD) and cross direction (CD), stretch, etc.
Different headbox control schemes should be chosen according to the different headbox types and stock delivery systems.
F1-Exit F2-Single F3-Cascade F4-Next

Figure 5.7.1(a) Headbox control system information in IAC

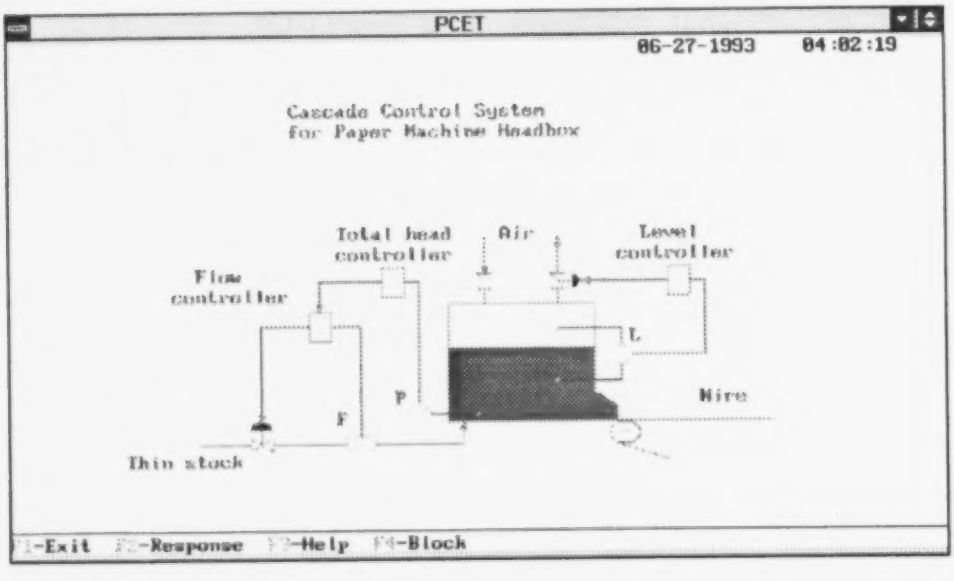


Figure 5.7.1(b) Cascade control system configuration in IAC

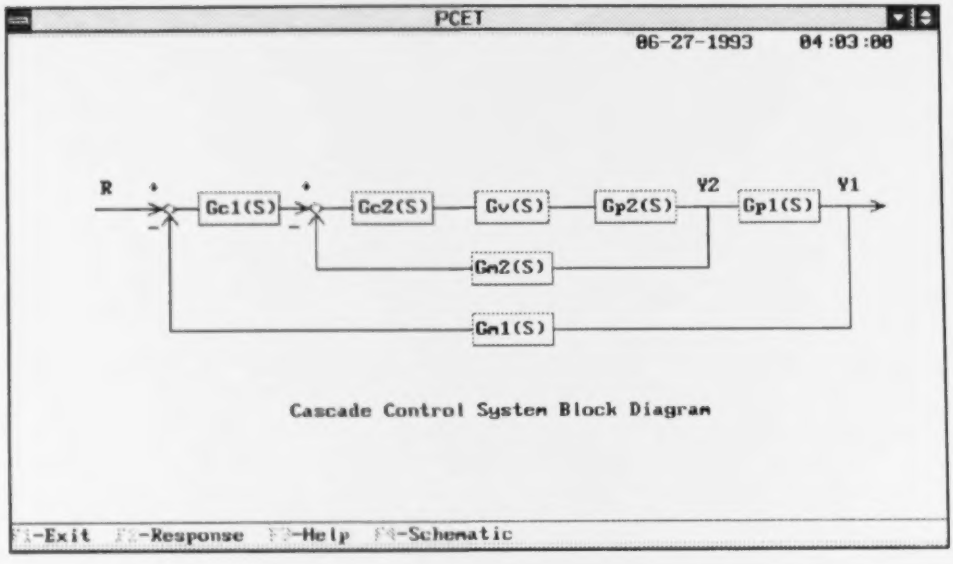


Figure 5.7.1(c) Cascade control system block diagram in IAC

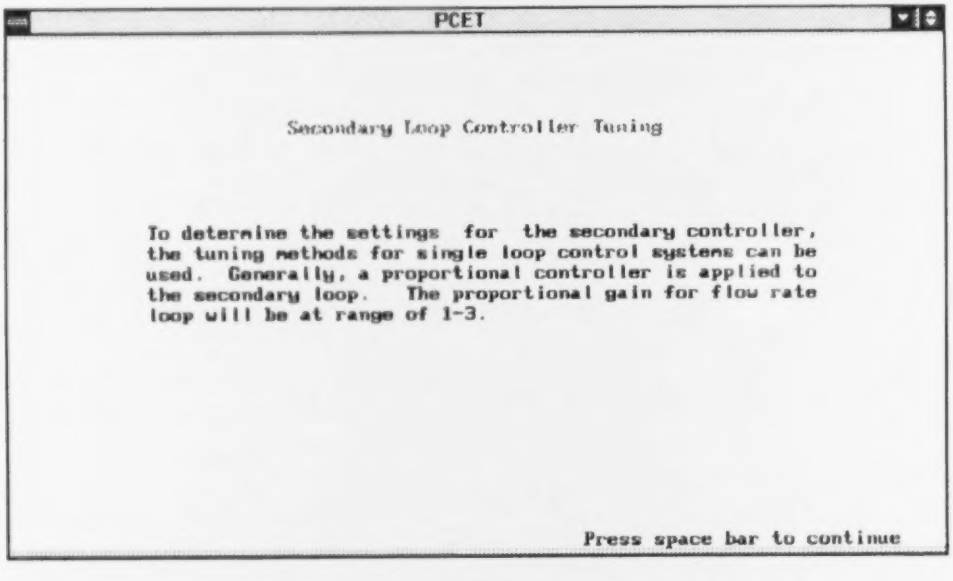

PCET
Secondary Loop Controller Tuning
To determine the settings for the secondary controller, the tuning methods for single loop control systems can be used. Generally, a proportional controller is applied to the secondary loop. The proportional gain for flow rate loop will be at range of 1-3.
Press space bar to continue

Figure 5.7.1(d) Primary loop controller tuning information

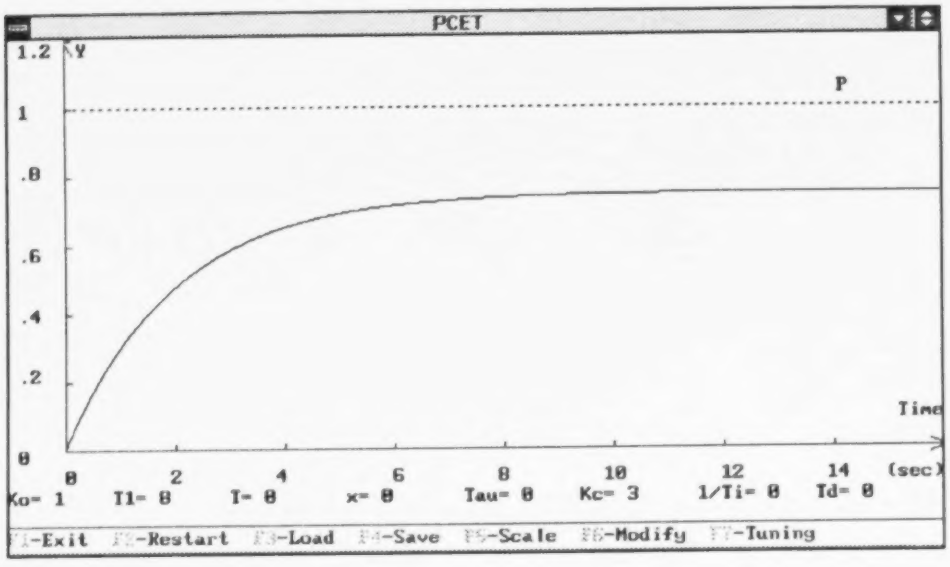


Figure 5.7.1(e) Secondary loop response of Example 5.7

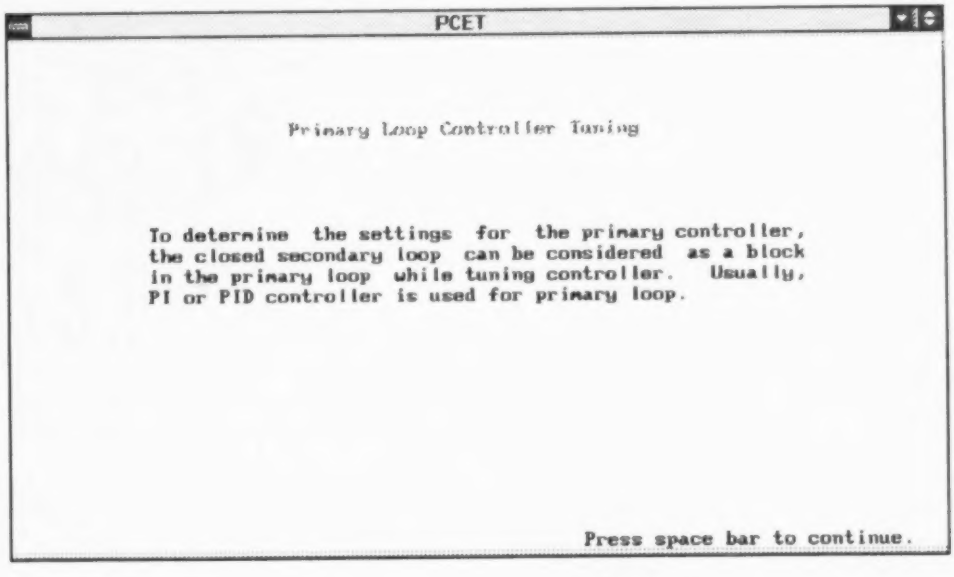

PCET
Primary Loop Controller Tuning
To determine the settings for the primary controller,
the closed secondary loop can be considered as a block
in the primary loop while tuning controller. Usually,
PI or PID controller is used for primary loop.
Press space bar to continue.

Figure 5.7.1(f) Secondary loop controller tuning information

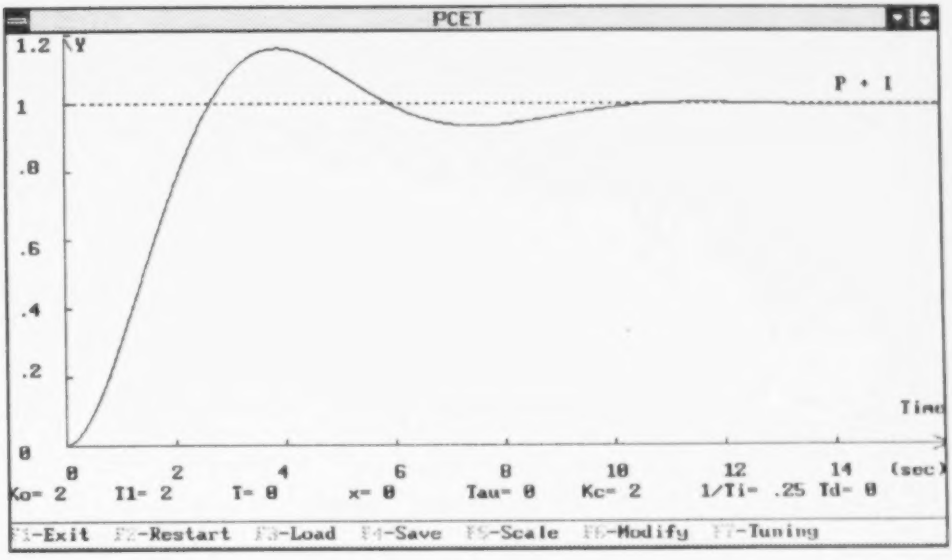


Figure 5.7.1(g) Primary loop response of Example 5.7

Chapter 6

General Discussion and Conclusions

This chapter contains some general concluding remarks. Conclusions specific to each project are given at the end of its respective chapter.

6.1 Results

In the project of process modelling by using the industrial sample data and historical information from ESSO Strathcona Refinery, an Artificial Neural Network (ANN) has been applied to predict the refinery product volatility. In addition, a Connection Weight Interpretor (CWI) has been also built to evaluate the relative significant contribution of input variables in the process model. For the integration of ANN as well as CWI, as subsystems for numerical computation, into Intelligent

Distributed Intelligent System (IDIS), Meta-COOP, a new architecture for a meta-system implementation, has been utilized. Through the indirect communication interface, ANN, CWI and/or other expert systems can swap their data interactively, for intelligent process modelling or operation, under the control of the meta-system. As the result of applying a multilayer feedforward Backpropagation (BP) neural network to inferential process modelling, it has been demonstrated that the ANN can model the product volatility at refinery plants. The ANN model can then be used for inferred volatility control. In this study, a couple of case studies has been carried out for two different processes using the BP network. In the first case study, inferential model for the volatility of Light Lube Oil (LLO) was developed by ANN with simulation data for Vacuum Pipe Still (VPS) in ESSO's Sarnia Refinery. Then, the ANN and Regression Analysis (RA) models were compared in representing the relation between the process stream data and the LLO volatility. The results showed that the ANN model matches the data much better than the RA model. Consequently, the ANN model was superior to the RA model at predicting the LLO volatility. In the second case study, ANN was applied to Atmospheric Pipe Still (APS) unit at ESSO's Strathcona Refinery for the prediction of Bottom Atmospheric Gas Oil (BAGO) volatility from noisy plant stream data. The results showed the ANN model to be able to fit even noisy plant data and to predict the BAGO volatility with a certain degree of accuracy.

The integration strategy to design an interactive computer-aided control system design environment for chemical processes was also presented, in which

several independently developed subprograms are coordinated by a meta-system. By this kind of integration approach, each subprogram can be easily modified without affecting other subprograms. It also allows other subprograms to be added to the system easily. Moreover, individual subprograms can be executed separately as a stand-alone system. The interactive graphics software package, PCET, has a hierarchical structure of several independently developed subprograms which are integrated by a meta-system. In the software package PCET, the meta-system serves as an interface mechanism to communicate individual software programs, and to build man-machine interface. Currently, PCET, written in the BASIC programming language for IBM PC's and compatibles, covers a wide spectrum of process control engineering tasks including Time Domain Analysis (TDA), Routh Stability Criterion (RSC), Root Locus Technique (RLT), Frequency Domain Analysis (FDA), Discrete-time System Analysis (DSA), Linear State Space Analysis (LSA) and Industrial Application Case (IAC). This software can be used to design control systems. It can also improve understanding of the basics of process control engineering, and can help users to gain practical experience on simulation and computer-aided design.

6.2 Contributions

The author feels that the following contributions were made both in academic research and industrial applications during my thesis research project.

1. The process modelling via ANN was developed for a real industrial process. All the data and knowledge about the industrial process were collected from ESSO's Strathcona Refinery. Evaluation by the ESSO Strathcona Refinery personnel was satisfactory, and the project was appreciated by industrial engineers. Such a project aims at narrowing the gap between academic research and real industrial applications.

2. PCET (Process Control Engineering Teachware) was developed for education purpose. The objective of PCET is to provide users with better understanding of process control system and design, as well as to train them. It is a menu-driven system running under a user-friendly and manual-free environment.

3. Through this thesis research, the meta-system, a new technology for integration issue, was investigated and utilized to integrate each independently developed susbsystem as a stand-alone system. This meta-system concept will pave the way for a new research direction for constructing IDIS.

Bibliography

Arkun, Y., V. Manousiouthakis, A. Palazoglu and V. Guruswamy, "Computer-Aided Analysis and Design of Robust Multivariable Control Systems for Chemical Processes," Computers & Chemical Engineering, Vol.9, No.1, p.27, 1985.

Astrom, K. J., "Process Control -- Past, Present, and Future," IEEE Control System Magazine, p.3, 1985.

Bridgeland, D. M. and M. N. Huhans, "Distributed Truth Maintenance," Proceedings of 8th National Conference on Artificial Intelligence, AAAI-90, p.72, 1990.

Buchanan, B. G., "Expert Systems in an Overview of Automated Reasoning and related fields," Journal of Automation Reasoning, Vol.1, p.28, 1985.

Buxton, B., "Impact of Packages Software for Process Control on Chemical Engineering Education and Research," Chemical Engineering Education, p.144, Summer 1985.

Caudill, M., "Avoiding the Great Backpropagation Trap," AI Expert, p.28, July 1991.

Cooper, D. J., L. Megan, and R. F. Hinde, Jr., "Comparing Two Neural Networks for Pattern Based Adaptive Process Control," AIChE Journal, Vol.38, p.41, 1992

.

Conry, S. E., D. J. MacIntosh and R. A. Meyer, "DARES: a Distributed Automated Reasoning System," Proceedings of 8th National Conference on Artificial Intelligence, AAAI-90, p.78, 1990.

Donat, J. S., N. Bhat and T. J. McAvoy, "Optimizing Neural Net based Predictive Control," Proceedings of 1990 American Control Conference, San Diego, CA., Vol.3, p.2466, 1990.

Garson, G. D., "Interpreting Neural-Network Connection Weights," AI Expert, p.47, April 1991.

Haesloop, D. and B. R. Holt, "A Neural Network Structure for System Identification," Proceedings of 1990 American control Conference, San Diego, CA., Vol.3, p.2460, 1990.

Hernandez, E. and Y. Arkun, "Neural Network Modeling and an Extended DMC Algorithm to Control Nonlinear Systems," Proceedings of 1990 American Control

Conference, San Diego, CA., Vol.3, p.2454, 1990.

Jacobstein, N., C. T. Kitzmiller and J. S. Kowalik, "Integrating Symbolic and Numeric Methods in Knowledge-based Systems: Current Status, Future Prospects, Driving Events, in Coupling Symbolic and Numerical Computing In Expert Systems, II," New York, NY., Elsevier Science Publishers B.V., p.3, 1988.

Kim, H. C., X. Shen and M. Rao, "Artificial Neural Network Approach to Inferential Control of Volatility in Refinery Plants," IFAC Workshop on Algorithms and Architectures for Real-Time Control, Seoul, Korea, p.135, August 31-September 2, 1992.

Kim, H. C., X. Shen, M. Rao, A. McIntosh and V. Mahalec, "Refinery Product Volatility Prediction using Neural Network," Prodeedings of the 42nd Canadian Chemical Engineering Conference, Toronto, Ontario, Canada, p.243, October 18-21, 1992.

Kim, H. C., H. Qiu and M. Rao, "Integrated Design Environment for Process Control," Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design, Napa, California, p.90, 1992.

Kitzmiller, C. T. and J. S. Kowalik, "Coupling Symbolic and Numeric Computing in Knowledge-based Systems," AI Magazine, p.85, Summer 1987.

Kowalik, J. S., "Coupling Symbolic and Numerical Computing in Expert Systems," Elsevier, Netherlands, 1986.

Leonard, J. and M. A. Kramer, "Improvement of the Backpropagation Algorithm for training Neural Networks," Computers & Chemical Engineering, Vol.14, p.337, 1990.

Mansour, M., M. Rimvall and W. Schaufelberger, "Computer Aided Design of Control Systems, An Integrated Approach," IFAC Computer Aided Design in control and Engineering Systems, Lyngby, Denmark, p.27, 1985.

Maybury, M. T., "Intelligent Multimedia Interfaces," IEEE Expert, p.875, February 1992.

Pang, G. K. H. and A. G. J. MacFarlane, "An Expert System Approach to Computer-Aided Design of Multivariable Systems," Springer-Verlag, New York, 1987.

Pang, G. K. H., M. Vidyasagar and A. J. Heunis, "Development of a New Generation of Interactive CACSD Environments," IEEE Control Systems Magazine, p.40, August 1990.

Rao, M., T. S. Jiang and J. P. Tsai, "A Framework of Integrated Intelligent

Systems," Proceedings of IEEE International Conference on System, Man and Cybernetics, Alexandria, Virginia, p.1133, 1987.

Rao, M., T. S. Jiang and J. P. Tsai, "IDSCA : An Intelligent Direction Selector for the Controller's Action in Multiloop Control Systems," International Journal of Intelligent Systems, Vol.3, p.361, 1988.

Rao, M., T. S. Jiang and J. P. Tsai, "Integration Strategy for Distributed Intelligent Systems," Journal of Intelligent and Robotic Systems, Vol.3, p.131, 1990.

Rao, M., "Lecture Note Series in Control and Information Science : Integrated System for Intelligent Control," Springer-verlag, 1991.

Rao, M. and H. Qiu, "Software for Teaching Process Control," IFAC Advances in Control Education, Boston, MA, p.118, 1991.

Rao, M., H. Qiu, H. C. Kim, Y. Ying and H. Zhou, "PC-Oriented Software for Process Control Education," International Journal of Applied Engineering Education, Vol.8, No.3, p.216, 1992.

Rao, M., H. Qiu, H. C. Kim and Y. Ying, "Integrating Industrial Application in Engineering Education," Proceedings of the 42nd Annual Canadian Chemical Engineering Conference, Toronto, Ontario, Canada, p.387, October 18-21, 1992.

Rao, M. and H. Qiu, "Process Control Engineering," Gordon and Breach Publisher Inc., Langhorne, PA, 1993.

Reklaitis, G. V., A. Ravindran and K. M. Ragsdell, "Engineering Optimization Methods and Application," John Wiley and Sons, New York, 1983.

Rumelhart, D. E. and J. L. McClelland, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition: 1. Foundations," MIT Press, Cambridge, 1986.

Sang, Z. T., M. Rao and T. W. Weber, "A Microcomputer-Based Simulation Laboratory for Process Control," Proceedings of SCS Multiconference, Application of Microcomputer Simulation, San Diego, CA., p.213, 1986.

Seborg, D. L., T. F. Edgar and D. A. Mellichamp, "Process Dynamics and Control," John Wiley & Sons, New York, 1989.

Snyder, W. and M. E. Hanyak, "Computer-Assisted Laboratory Stations," Chemical Engineering Education, p.26, Winter 1985.

Soucek, Branko and The IRIS Group, "Neural and Intelligent Systems Integration: Fifth and Six Generation Integrated Reasoning Information Systems," Wiley, New York, 1991.

Spang, H. A., "Experience and Future Needs in Computer-Aided Control Design," IEEE Control System Magazine, p.18, February 1985.

Stephanopoulos, G., "Chemical Process Control," Prentice-Hall, Inc., 1984.

Touretzky, D. S., "Advances in Neural Information Processing Systems," Morgan Kaufmann, California, 1990.

Vemuri, V., "Artificial Neural Networks : Theoretical Concepts," Computer Society press of the IEEE, Washington D. C., 1988.

Wasserman, P. D., "Neural Computing Theory and Practice," Van Nostrand Reinhold, New York, 1989.

Watanabe, K., I. Matsuura, M. Abe, M. Kubota and D. M. Himmeblau, "Incipient Fault Diagnosis of Chemical Processes via Artificial Neural Networks", AIChE Journal, Vol.35, No.11, p.1903, 1989.

Wong, F. S., W. Dong and M. Blanks, "Coupling of Symbolic and Numerical Computations on a Microcomputer," AI in Engineering, Vol. 3, p.32, 1988.

Appendix A

Source Code for ANN

```
C       BACKPROPAGATION NEURAL NETWOK
C       WITH CONJUGATE GRADIENT IN TRAINING
C       BY HEON CHANG KIM
C
C
C
        PROGRAM ANNCG

        REAL X(420),WI(20,20),WJ(1,20),G(420),W(10000)
        REAL XN(300,20),YM(300,20),YP(300,1),YD(300,1)
        CHARACTER A72(8)*10,FNAME*50
        CHARACTER*16 CPN(30),PCN(30)
        COMMON IP,MY,NX,WI,WJ,IN,XN,YM,YP,YD
        COMMON /STAT/ SSTO,RVALUE
        EXTERNAL FUNCT

        OPEN(7,FILE='EXNET.DAT',STATUS='OLD')
```

```
      READ(7,*)ID
      CLOSE(7)

      WRITE(*,1)

100   WRITE(*,2)
      READ(*,'(A)') FNAME
      ICH = INDEX(FNAME,' ')-1
      IPT = INDEX(FNAME,'.')-1
      IF (ICH .GT. 0) THEN
        IF (IPT .LT. 0) THEN
          WRITE(A72(1)(1:12),3) ICH
          IPT = ICH
          WRITE(FNAME,A72) FNAME
        ENDIF
        OPEN(5,FILE=FNAME,STATUS='OLD',ERR=110)
        WRITE(A72(1)(1:5),4) IPT
        WRITE(PCN(1)(1:50),A72) FNAME
        CPN(1)=FNAME
        GO TO 120
110     WRITE(A72(1)(1:33),5)INDEX(FNAME,' ')
        WRITE(*,A72) FNAME
      ENDIF
      GO TO 100

120   WRITE(A72(1)(1:36),6) IPT
      WRITE(*,A72) PCN(1)
      READ(*,'(A)') FNAME
      IF (INDEX(FNAME,'.') .LT. 1) THEN
        ICH = INDEX(FNAME,' ')-1
        IF (ICH .LE. 0) THEN
```

```
          ICH = IPT
          WRITE(A72(1)(1:5),4) IPT
          WRITE(FNAME,A72) PCN(1)
        ENDIF
        WRITE(A72(1)(1:12),7) ICH
        WRITE(FNAME,A72) FNAME
      ENDIF
      CPN(10)=FNAME
      OPEN(6,FILE=FNAME,STATUS='NEW',ERR=130)
      GO TO 140
130   WRITE(A72(1)(1:51),8)INDEX(FNAME,'  ')
      WRITE(*,A72) FNAME
      READ(*,'(A)') A72(1)
      IF (A72(1).EQ.'Y' .OR. A72(1).EQ.'Y') THEN
        OPEN(6,FILE=FNAME,MODE='WRITE')
      ELSEIF(A72(1).EQ.'N' .OR. A72(1).EQ.'N') THEN
        WRITE(*,*)
        GO TO 120
      ELSE
        GO TO 130
      ENDIF

140   WRITE(A72(1)(1:12),9) ICH
      WRITE(FNAME,A72) PCN(1)
      CALL GETDAT(IYR,IMON,IDAY)

      WRITE(A72(1)(1:29),10) INDEX(CPN(1),'  '),INDEX(CPN(10),'  ')
      WRITE(6,A72) CPN(1),CPN(10)

1     FORMAT(/,10X,'+',52('-'),'+',
     &  /,10X,'¦ BACKPROPAGATION NEURAL NETWORK VERSION 1.00,  1992 ¦',
```

```
     &     /,10X,'|                    BY HEON CHANG KIM                    |',
     &     /,10X,'+',52('-'),'+',/)
2          FORMAT(1X,'INPUT  FILE [.NET]: ',\)
3          FORMAT('(A',I2,',',7H'.NET'))
4          FORMAT('(A',I2,')')
5          FORMAT('(/,29X,A',I2,21H' DOES NOT EXIST.',/))
6          FORMAT(21H(1X,'OUTPUT FILE [',A,I2,13H,'.OUT]: ',\))
7          FORMAT('(A',I2,',',7H'.OUT'))
8          FORMAT('(/,29X,A',I2,38H' ALREADY EXISTS. OVERWRITE ? (Y/N) ',
     &           ,'\)')
9          FORMAT('(A',I2,',',5H'.W'))
10         FORMAT(11H(' DRN: ',A,I2,13H,/,' PRN: ',A,I2,')')
11         FORMAT(8A10)
12         FORMAT(/,40X,'DATE       ',I2,'/',I2.2,'/',I2.2,/)


           READ(5,*)IN,NX,MY,IP,ACC,DFPRED

           READ(5,*)((XN(J,I),I=1,NX),(YD(J,I),I=1,IP),J=1,IN)

           COD=0
           SSTO=0
           SUM=0
           DO 987 J=1,IN
              DO 987 I=1,IP
                 SSTO=SSTO+YD(J,I)*YD(J,I)
                 SUM=SUM+YD(J,I)
987        CONTINUE
           SSTO=SSTO-SUM*SUM/IN
           SUM=0
```

```
      TR=0

      IF(ID .EQ. 2) THEN
        OPEN(7,FILE='EXNET.DAT',MODE='WRITE')
        WRITE(7,'(A)')FNAME
        CLOSE(7)
      ENDIF

      IF(ID .NE. 0) THEN
        OPEN(7,FILE=FNAME,STATUS='OLD')
        READ(7,*)((WI(I,K),K=1,NX),I=1,MY)
        READ(7,*)((WJ(I,J),J=1,MY),I=1,IP)
        CLOSE(7)
        GOTO 1000
      ENDIF

      DO 150 I=1,IP
        DO 150 J=1,MY
          S=3.141592/(I+J-1)
          WJ(I,J)=SIN(S)
150   CONTINUE

      DO 200 I=1,MY
        DO 200 J=1,NX
          S=3.141592/(I+J-1)
          WI(I,J)=COS(S)
200   CONTINUE

1000  DO 300 I=1,IP
        DO 300 J=1,MY
          X(J)=WJ(I,J)
300   CONTINUE
```

```
      DO 400 I=1,MY
        DO 400 J=1,NX
          X(MY+J+NX*(I-1))=WI(I,J)
400   CONTINUE

      IF(ID .EQ. 1) GOTO 850

      N=(NX+1)*MY
      MAXFN=0

999   CALL ZXCGR(FUNCT,N,ACC,MAXFN,DFPRED,X,G,F,W,IER,NCALLS)

1100  CALL XW(X)

      WRITE(6,*)'TRAINING TIME =', NCALLS
      WRITE(6,*)IN,NX,MY,IP,ACC,DFPRED,F

      OPEN(7,FILE=FNAME,MODE='WRITE')
      WRITE(7,*)NX,MY,IP
      WRITE(7,*)((WI(I,K),K=1,NX),I=1,MY)
      WRITE(7,*)
      WRITE(7,*)((WJ(I,J),J=1,MY),I=1,IP)

      WRITE(6,*)'********** TRAINING ***********'
600   SSE=0

      IF(ID .EQ. 1) OPEN(7,FILE='EXNET.DAT',MODE='WRITE')
      DO 700 II=1,IN
        CALL XNYP3(II)
        WRITE(6,*)II,(YD(II,I),YP(II,I),YP(II,I)-YD(II,I),I=1,IP)
          IF(ID .EQ. 1) WRITE(7,*)YP(II,I)
```

```
      DO 700 I=1,IP
         SSE=SSE+(YP(II,I)-YD(II,I))**2
700   CONTINUE
      RVALUE=1-SSE/SSTO
      WRITE(6,*)'********** R SQUARE =',RVALUE,'      RMS =',SSE/IN

      STOP

      READ(5,*)NP
      READ(5,*)((XN(J,I),I=1,NX),(YD(J,I),I=1,IP),J=1,NP)

      SSTO=0
      SUM=0
      DO 800 J=1,NP
         DO 800 I=1,IP
            SSTO=SSTO+YD(J,I)*YD(J,I)
            SUM=SUM+YD(J,I)
800   CONTINUE
      SSTO=SSTO-SUM*SUM/NP
      IN=NP
      COD=1
      WRITE(6,*)'********** PREDICTION ***********'
      GOTO 600

      STOP
      END



      SUBROUTINE FUNCT(N,X,F,G)

      REAL X(420),WI(20,20),WJ(1,20),G(420)
```

```
      REAL XN(300,20),YM(300,20),YP(300,1),YD(300,1)
      COMMON IP,MY,NX,WI,WJ,IN,XN,YM,YP,YD

      CALL XW(X)

      F=0.
      DO 10 II=1,IN
         CALL XNYP3(II)
         F=F+.5*(YD(II,1)-YP(II,1))**2.
10    CONTINUE

      CALL DFDX(X,G)

      RETURN
      END



      SUBROUTINE DFDX(X,G)

      REAL X(420),WI(20,20),WJ(1,20),G(420)
      REAL XN(300,20),YM(300,20),YP(300,1),YD(300,1)
      COMMON IP,MY,NX,WI,WJ,IN,XN,YM,YP,YD

      DO 10 J=1,MY
         G(J)=0.
         DO 10 K=1,IN
            G(J)=G(J)+(YP(K,1)-YD(K,1))*YP(K,1)*(1-YP(K,1))
     &                 *YM(K,J)
10    CONTINUE

      DO 20 I=1,MY
```

```
      DO 20 J=1,NX
        SUM=0.
        DO 120 K=1,IN
          SUM=SUM+(YP(K,1)-YD(K,1))*YP(K,1)*(1-
     &  YP(K,1))*YM(K,I)*(1-YM(K,I))*WJ(1,I)*XN(K,J)
120     CONTINUE
        G(MY+J+NX*(I-1))=SUM
20    CONTINUE

      RETURN
      END



      SUBROUTINE XNYP3(II)

      REAL WI(20,20),WJ(1,20)
      REAL XN(300,20),YM(300,20),YP(300,1),YD(300,1)
      COMMON IP,MY,NX,WI,WJ,IN,XN,YM,YP,YD

      DO 100 I=1,MY
        SUM=0.
        DO 10 J=1,NX
10        SUM=SUM+WI(I,J)*XN(II,J)
        YM(II,I)=1/(1+EXP(-SUM))
100   CONTINUE

      DO 200 I=1,IP
        SUM=0.
        DO 20 J=1,MY
20        SUM=SUM+WJ(I,J)*YM(II,J)
        YP(II,I)=1./(1+EXP(-SUM))
```

```
200     CONTINUE

        RETURN
        END



        SUBROUTINE XW(X)

        REAL X(420),WI(20,20),WJ(1,20)
        COMMON IP,MY,NX,WI,WJ

        DO 300 I=1,IP
          DO 300 J=1,MY
            WJ(I,J)=X(J)
300     CONTINUE

        DO 400 I=1,MY
          DO 400 J=1,NX
            WI(I,J)=X(MY+J+NX*(I-1))
400     CONTINUE

        RETURN
        END



        SUBROUTINE ZXCGR(FUNCT,N,ACC,MAXFN,DFPRED,X,G,F,W,IER,NCALLS)
     &
        INTEGER  N,MAXFN,IER
        REAL     ACC,DFPRED,X(N),G(N),F,W(1)
        INTEGER MAXLIN,MXFCON,I,IGINIT,IGOPT,IRETRY,IRSDG,
```

```
     &          IRSDX,ITERC,ITERFM,ITERRS,IXOPT,NCALLS,NFBEG,
     &            NFOPT
          REAL BETA,DDSPLN,DFPR,FCH,FINIT,FMIN,GAMDEN,GAMA,
     &       GINIT,GMIN,GNEW,GSPLN,GSQRD,SBOUND,STEP,STEPCH,
     &         STMIN,SUM,WORK
          COMMON /STAT/ SSTO,RVALUE
          DATA      MAXLIN/5/,MXFCON/2/

          IER = 0
          IRSDX = N
          IRSDG = IRSDX+N
          IGINIT = IRSDG+N
          IXOPT = IGINIT+N
          IGOPT = IXOPT+N
          ITERC = 0
          NCALLS = 0
          ITERFM = ITERC

5         NCALLS = NCALLS+1

          CALL FUNCT (N,X,F,G)

          IF (NCALLS.GE.2) GO TO 20

10        DO 15 I=1,N
15        W(I) = -G(I)

          ITERRS = 0

          IF (ITERC.GT.0) GO TO 80
20
          GNEW = 0.0
```

```
      SUM = 0.0

      DO 25 I=1,N
         GNEW = GNEW+W(I)*G(I)
25    SUM = SUM+G(I)**2

      RVALUE=1-2*F/SSTO

      WRITE(*,*)NCALLS,F,SUM,RVALUE

      IF (NCALLS.EQ.1) GO TO 35

      FCH = F-FMIN

      IF (FCH) 35,30,50
30    IF (GNEW/GMIN.LT.-1.0) GO TO 45

35    FMIN = F
      GSQRD = SUM
      NFOPT = NCALLS

      DO 40 I=1,N
         W(IXOPT+I) = X(I)
40    W(IGOPT+I) = G(I)

45    IF (SUM.LE.ACC) GO TO 9005

50    IF (NCALLS.NE.MAXFN) GO TO 55

      IER = 131

      GO TO 9000
```

```
55      IF (NCALLS.GT.1) GO TO 100

        DFPR = DFPRED
        STMIN = DFPRED/GSQRD

80      ITERC = ITERC+1

        FINIT = F
        GINIT = 0.0

        DO 85 I=1,N
          W(IGINIT+I) = G(I)
85      GINIT = GINIT+W(I)*G(I)

        IF (GINIT.GE.0.0) GO TO 165

        GMIN = GINIT
        SBOUND = -1.0
        NFBEG = NCALLS
        IRETRY = -1

        STEPCH = AMIN1(STMIN,ABS(DFPR/GINIT))

        STMIN = 0.0

90      STEP = STMIN+STEPCH
        WORK = 0.0

        DO 95 I=1,N
          X(I) = W(IXOPT+I)+STEPCH*W(I)
95      WORK = AMAX1(WORK,ABS(X(I)-W(IXOPT+I)))
```

```
      IF (WORK.GT.0.0) GO TO 5
      IF (NCALLS.GT.NFBEG+1) GO TO 115
      IF (ABS(GMIN/GINIT)-0.2) 170,170,115

100   WORK = (FCH+FCH)/STEPCH-GNEW-GMIN
      DDSPLN = (GNEW-GMIN)/STEPCH

      IF (NCALLS.GT.NFOPT) SBOUND = STEP
      IF (NCALLS.GT.NFOPT) GO TO 105
      IF (GMIN*GNEW.LE.0.0) SBOUND = STMIN

      STMIN = STEP
      GMIN = GNEW
      STEPCH = -STEPCH

105   IF (FCH.NE.0.0) DDSPLN = DDSPLN+(WORK+WORK)/STEPCH
      IF (GMIN.EQ.0.0) GO TO 170
      IF (NCALLS.LE.NFBEG+1) GO TO 120
      IF (ABS(GMIN/GINIT).LE.0.2) GO TO 170
110   IF (NCALLS.LT.NFOPT+MAXLIN) GO TO 120

115   IER = 129
      GO TO 170

120   STEPCH = 0.5*(SBOUND-STMIN)

      IF (SBOUND.LT.-0.5) STEPCH = 9.0*STMIN

      GSPLN = GMIN+STEPCH*DDSPLN

      IF (GMIN*GSPLN.LT.0.0) STEPCH = STEPCH*GMIN/(GMIN-GSPLN)
```

```
      GO TO 90

125   SUM = 0.0

      DO 130 I=1,N
130   SUM = SUM+G(I)*W(IGINIT+I)

      BETA = (GSQRD-SUM)/(GMIN-GINIT)

      IF (ABS(BETA*GMIN).LE.0.2*GSQRD) GO TO 135

      IRETRY = IRETRY+1

      IF (IRETRY.LE.0) GO TO 110
135   IF (F.LT.FINIT) ITERFM = ITERC
      IF (ITERC.LT.ITERFM+MXFCON) GO TO 140

      IER = 132
      GO TO 9000

140   DFPR = STMIN*GINIT

      IF (IRETRY.GT.0) GO TO 10
      IF (ITERRS.EQ.0) GO TO 155
      IF (ITERC-ITERRS.GE.N) GO TO 155
      IF (ABS(SUM).GE.0.2*GSQRD) GO TO 155

      GAMA = 0.0
      SUM = 0.0

      DO 145 I=1,N
        GAMA = GAMA+G(I)*W(IRSDG+I)
```

```
145      SUM = SUM+G(I)*W(IRSDX+I)

         GAMA = GAMA/GAMDEN

         IF (ABS(BETA*GMIN+GAMA*SUM).GE.0.2*GSQRD) GO TO 155

         DO 150 I=1,N
150      W(I) = -G(I)+BETA*W(I)+GAMA*W(IRSDX+I)

         GO TO 80

155      GAMDEN = GMIN-GINIT

         DO 160 I=1,N
            W(IRSDX+I) = W(I)
            W(IRSDG+I) = G(I)-W(IGINIT+I)
160      W(I) = -G(I)+BETA*W(I)

         ITERRS = ITERC
         GO TO 80

165      IER = 130

170      IF (NCALLS.EQ.NFOPT) GO TO 180

         F = FMIN

         DO 175 I=1,N
            X(I) = W(IXOPT+I)
175      G(I) = W(IGOPT+I)

180      IF (IER.EQ.0) GO TO 125
```

```
9000    CONTINUE


9005    RETURN
        END
```

Appendix B

Meta-knowledgebase Source code for Integration of ANN and CWI into IDIS

```
/
*****************************************************************************/
/*        Module              : Answer.fra                                   */
/*        Function            : Design Answer Unit                           */
/*                                                                           */
/
*****************************************************************************/
globe
Answer1: Answer;
instance: ANN;
procedure: String;
End

Unit: Answer in_knowledge_base Answer.kbs
```

```
Memberslot: index from Answer
   Inheritance: Override.Values
   Valueclass: integer
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: Unknown

/
*****************************************************************************/
/*        Module                : ANN.fra                                    */
/*        Function              : Design ANN Unit                            */
/*                                                                           */
/
*****************************************************************************/
Unit: ANN in_knowledge_base Answer.kbs

Memberslot: procedure from ANN
   Inheritance: Override.Values
   Valueclass: string
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: Unknown

Memberslot: question from ANN
   Inheritance: Override.Values
   Valueclass: RULES
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: {
     rule 1
       fact Answer1.index=2
```

```
        then _FRAME.procedure:=''train''
      rule 2
        fact Answer1.index=1
        then _FRAME.procedure:=''predict''
          }

Memberslot: prediction from ANN
   Inheritance: Override.Values
   Valueclass: real
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: Unknown

Memberslot: decide from ANN
   Inheritance: METHOD
   Valueclass: METHODS
   Cardinality.Min: 1
   Cardinality.Max: 1
   Values: decision

METHOD decision(design:keyword)
VAR
   design: keyword;
   xchar: string;
   yout:real;
   nxin:integer;
   w,r: file;
BEGIN
   writeln(‘‘’’);
   writeln(‘‘’’);
   write(‘‘Please enter’’);
   writeln(‘‘ 1 or 2.’’);
```

```
writeln(''    1. Prediction'');
writeln("     2. Training");
writeln('''');
writeln('''');
reason(_FRAME,''question'');
 open(w,"exnet.dat","w");
writeln(w,Answer1.index);
close(w);
system(''ann-conj'');
if _FRAME.procedure=''train''  then
  begin
    system('ann-cwi');
   end;
if (_FRAME.procedure=''predict'') then
  begin
   open(r,xchar,"r");
   readln(r,yout);
   _FRAME.prediction=yout;
   write(''The predicted '');
   write(''quality is '');
   writeln(yout);
  end;

END.
```

Appendix C

PCET Evaluation Form

EVALUATION OF PCET

Please provide us with your thoughtful responses to the following questions. We wish to improve the quality of our Computer-Aided Control System Design package, PCET, by monitoring your valuable opinion.

USER CHARACTERISTICS

1. What is your university year?

   1. 1st 2. 2nd 3. 3rd 4. 4th 5. Post-degree

2. What is your age?

   1. under 25 2. between 25 and 30

   3. between 30 and 35 4. over 35

3. What is your major? If your choice is 5, please specify.

1. Process Control
2. Chemical Engineering
3. Electrical Engineering
4. Mechanical Engineering
5. Others :

4. Have you any industrial experience? If yes, please provide job description?

1. no
2. less than 2 years
3. between 2 and 5 years
4. more than 5 years

5. What kind of computer are you familiar with? If your choice is 5, please specify.

1. IBM PC
2. Machintoshi
3. workstation
4. mainframe
5. others

6. What kind of programming language are you familiar with? If your choice is 7, please specify.

1. Assembly
2. Basic
3. Fortran
4. C
5. C++
6. LISP
7. others

7. How often did you meet with instructor or TA regarding to PCET?

1. never
2. one time
3. two times
4. more than three times

8. Have you ever used other control system design packages? If your choice is 5, please specify.

1. no
2. PCMATLAB
3. PROGRAM CC
4. TUTSIM

5. others

LEARNING

9. How long did it take for you to get used to PCET?

1. less than 10 minutes 2. between 10 and 30 minutes

3. between 30 and 60 minutes 4. more than 1 hour

10. Did you gain a good understanding of concepts/principles/skills in designing control system by using PCET?

1. completely 2. much 3. some 4. little 5. not at all

11. Did you gaine practical experiences on simulation and computer-aided design of control system by using PCET?

1. completely 2. much 3. some 4. little 5. not at all

SATISFACTION

12. Did you like independent study?

1. absolutely 2. much 3. some 4. little 5. not at all

13. Have you had any difficulty in using PCET? If yes, please specify.

14. Was PCET helpful for designing control system?

1. absolutely 2. much 3. some 4. little 5. not at all

15. If you are to make an improvement on PCET, what function would you like to add?

16. Please comment on the strength/weakness of PCET.

OVERALL

17. Are you going to keep using PCET? Please provide reasons.

18. Are you going to recommend PCET to other colleagues? Please provide reasons.

19. How do you like to rate the Computer-Aided Control System Design package, PCET?

1. excellent 2. good 3. fair 4. poor 5. terrible